НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Теплоенергетичний факультет

Кафедра автоматизації проектування енергетичних процесів і систем

"На правах рукопису"
УДК ___004.02___

«До захисту допущено»
Завідувач кафедри
_____ О.В. Коваль
(підпис) (ініціали, прізвище)
“___”_____ 2019 р.

Магістерська дисертація

зі спеціальності 121 Інженерія програмного забезпечення
за спеціалізацією Інженерія програмного забезпечення розподілених систем
на тему “Проеціювання траекторії руху на об’єкти реального світу для мобільного додатку навігації з використанням доповненої реальності”

Виконав: студент 6 курсу, групи _ТВ-81мп_
Зарицький Віталій Петрович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доц., к.т.н. Гагарін О. О. _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019

Національний технічний університет України
“ Київський політехнічний інститут ім. Ігоря Сікорського”

Факультет теплоенергетичний

Кафедра автоматизації проектування енергетичних процесів і систем

Рівень вищої освіти другий, магістерський

зі спеціальності - 121 Інженерія програмного забезпечення
за спеціалізацією - Інженерія програмного забезпечення розподілених систем

ЗАТВЕРДЖУЮ
Завідувач кафедри
Коваль О.В. _____
(прізвище, ініціали) (підпис)
«____» _____ 2019р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ ДИСЕРТАЦІЮ СТУДЕНТУ

Зарицький Віталій Петрович
(прізвище, ім’я, по батькові)

1. Тема дисертації Проеціювання траекторії руху на об’єкти реального світу для мобільного додатку навігації з використанням доповненої реальності

Науковий керівник доц., к.т.н., Гагарін Олександр Олександрович
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “_04_” _листопада_2019 року №_3812-с_
2. Строк подання студентом дисертації _____9_грудня 2019_____
3. Об’єкт дослідження ___Засоби проеціювання траекторії руху в системах навігації

4. Предмет дослідження ___ Засоби інтерактивної взаємодії користувача в процесі пошуку шляху в системах навігації.

5. Перелік питань, які потрібно розробити_____
Розглянути методи навігації в будівлях; проаналізувати існуючі підсистеми проеціювання траекторії руху в додатках навігації приміщеннями; розробити підсистему проеціювання траекторії руху для розроблюваного додатку навігації всередині приміщення для смартфонів; реалізувати інтерфейс системи; інтегрувати модусь з іншими частинами програми; протестувати додаток._____

6. Орієнтований перелік ілюстративного матеріалу _____
__«Актуальність», «Мета та завдання роботи», «Функції користувача системи», «Взаємодія компонентів системи», «Засоби розробки», «NavMesh, або навігаційна сітка», «Схема роботи підсистеми», «Скриншоти роботи додатку», «Демонстрація роботи додатку», «Переваги і недоліки системи», «Висновки».

_____

_____

7. Орієнтований перелік публікацій _____

Проблема вибору раціонального методу позиціювання користувача для системи навігації (“Сучасні проблеми наукового забезпечення енергетики” XVII міжнародної науково-практичної конференції аспірантів, магістрів, студентів)

8. Дата видачі завдання «_28_»___вересня___2019р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання магістерської дисертації	Строки виконання етапів магістерської дисертації	Примітка
1	Отримання завдання	28.09.18 р.	
2	Опрацювання літературних джерел	01.10.18 р. – 03.02.19 р.	
3	Підготовка матеріалів дисертації	04.02 – 31.05.19 р.	
4	Підготовка доповідей на конференції	11.03 – 29.03.19 р.	
6	Розробка програмного продукту	03.06 – 25.10.19 р.	
5	Переддипломна практика	02.09 – 25.10.19 р.	
7	Захист програмного продукту	26.10.19 р.	
8	Розробка стартап-проекту	11.11 – 19.11.19 р.	
9	Передзахист	20.11.19 р.	
10	Оформлення дисертації	21.11- 29.11.19 р.	
11	Захист	16.12.19 р.	

Студент _____ _____Зарицький В.П._____
(підпис) (прізвище та ініціали)

Науковий керівник _____ _____Гагарін О.О_____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Структура і обсяг дипломної роботи

Магістрська дисертація складається зі вступу, шести розділів і висновку. Робота містить в собі 30 джерел за переліком посилань, 39 ілюстрацій, 23 таблиць. Крім того, в кінці розміщено 1 додаток. Основна частина роботи викладена на 92 сторінках.

Актуальність теми

Досить часто, перебуваючи у якійсь будівлі, особливо вперше, ми зіштовхуємося з проблемою: потрібно дістатися певного місця, при цьому не завжди зрозуміло, як це робити. Іноді біля входу може бути розміщений план поверху, який допоможе; також на стінах іноді розташовані вказівники з надписом аудиторії, показуючи, куди йти. Проте, звісно, такий варіант не є зручним.

Саме тому існує необхідність створення додатку навігації для мобільних телефонів. Така програма дозволить швидко знайти необхідне місце в будівлі, що зекономить час, наприклад, першокурсникам, шо вперше шукають якусь аудиторію.

AR — augmented reality — доповнена реальність — це доповнення фізичного світу додатковими даними (цифровими) з використанням пристроїв (наприклад, смартфонів) і програм. Простіше кажучи, доповнена реальність — це поєднання цифрового світу з фізичним і, таким чином, збільшення досвіду від реального світу. Саме тому її використання для подібного додатку, наприклад, для відображення попередньо знайденого оптимального шляху до пункту призначення, разом з user-friendly інтерфейсом зробить продукт не тільки більш інтерактивним, зручним і практичним, а ще і цікавим для користувача.

Метою дослідження є створення підсистеми для системи навігації для мобільних телефонів, що буде знаходити шлях до точки призначення і показувати його користувачеві за допомогою доповненої реальності.

Для досягнення поставленої мети потрібно виконати наступні **задачі дослідження**:

— проаналізувати існуючий процес навігації, в тому числі і в корпусі ТЕФ;

— розглянути схожі існуючі системи та порівняти їх з розроблюваною системою;

— розглянути варіанти використання доповненої реальності та обрати один найкращий;

— обрати раціональний метод або інструмент, що відповідатиме за пошук шляху (навігацію);

— на основі рішень попередніх задач реалізувати програмну компоненту.

Об’єкт дослідження: Засоби проеціювання траекторії руху в системах навігації.

Предмет дослідження: Засоби інтерактивної взаємодії користувача в процесі пошуку шляху в системах навігації.

Методи дослідження: Методи використання доповненої реальності в системах навігації, методи побудови архітектури програмних систем.

Наукова новизна:

Набуло подальшого розвитку використання доповненої реальності для відображення траекторії руху користувача в системах навігації, а особливо методу з фіксованою відносно користувача стрілкою, що своїм поворотом вказує шлях.

За рахунок розробленого додатку навігації удосконалено процес навігації в корпусі ТЕФ.

Практичне значення полягає в тому, що розроблений додаток, як передбачається, буде використовуватися студентами ТЕФ, в тому числі і в режимі камери з доповненою реальністю, що допоможе більш швидко, зручно та ефективно знаходити шлях до бажаної аудиторії.

Апробація. Результати досліджень стосовно цілої системи взагалом представлені на XVII міжнародній науково-практичній конференції аспірантів, магістрів, студентів на тему «Сучасні проблеми наукового забезпечення енергетики» 2019 року.

Ключові слова: НАВІГАЦІЯ, ДОПОВНЕНА РЕАЛЬНІСТЬ, ПОШУК ШЛЯХУ, ТРЕКТОРІЯ РУХУ, ПУНКТ ПРИЗНАЧЕННЯ, ДРУЖНІЙ ДО КОРИСТУВАЧА ІНТЕРФЕЙС, ARCORE.

ABSTRACT

The structure and volume of the graduate work.

The master's thesis consists of an introduction, six sections and a conclusion. The work contains 30 sources in the list of references, 38 illustrations, 23 tables. Also, 1 appendix is added at the end. The main part of the work contains 96 pages.

Actuality of theme.

Quite often, when we are in a building, especially for the first time, we face the problem: we need to get to a certain place, but it is not always clear how to do it. Occasionally a floor plan may be placed at the entrance that may help; sometimes there are also pointers on the walls that tell us where we should go. However, of course, this option is not convenient.

That is why there is a need to create a navigation application for mobile phones. Such a program will quickly find the right place in the building. This will save time, for example, for first-year student for the first time seeking a certain room.

AR - Augmented reality — is the augmentation of the physical world with additional data (digital) using devices (such as smartphones) and applications. Simply, augmented reality is a combination of the digital world with the physical, and thus an increase in experience from the real world. That is why using it for a similar application, for example, to display the previously found optimal path to the destination, together with the user-friendly interface will make the product not only more interactive, convenient and practical, but also interesting for the user.

The aim of the study is a creation of a subsystem of navigation system for smartphones that will find the way to destination and show it to the user with augmented reality.

To achieve this goal, there is a need to complete the following **research objectives**:

— analyze the existing navigation process, including the TEF department;

— examine similar existing systems and compare them with the system under development;

— consider methods of using augmented reality and choose the best one;

— choose a rational method or tool that will be responsible for finding the path (navigation)

— create the software component, basing on the solutions of the previous problems.

Object of study: Means of projection of motion trajectory in navigation systems.

Subject of study: Means of user interaction in the process of finding the way in navigation systems.

Research Methods: Methods of using augmented reality in navigation systems, methods of constructing software systems architecture.

Scientific novelty:

The use of augmented reality for displaying the trajectory of the user's movement in navigation systems has further evolved, and in particular the method with a fixed relative to the user arrow that indicates direction with its rotation.

Due to the developed navigation application, the navigation process in the TEF department has been improved.

The **practical significance** is that the developed application is expected to be used by TEF students, including in augmented reality camera mode, which will help find the desired room more quickly, conveniently and effectively.

Approbation. The results of the research on the whole system were presented at the 18th International Scientific and Practical Conference of post-graduate students, masters and students on the topic "Modern problems of scientific support of energy" in 2019.

Keywords: NAVIGATION, AUGMENTED REALITY, PATHFINDING, TRAJECTORY OF MOTION, DESTINATION, USER-FRIENDLY INTERFACE, ARCORE.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Навігація — це поле дослідження, яке фокусується на процесі спостереження та контролю руху судна, транспортного засобу чи людини з одного місця в інше; сам цей процес.

AR — augmented reality — доповнена реальність — це доповнення фізичного світу додатковими даними (цифровими) з використанням пристроїв (наприклад, смартфонів) і програм.

Zoom, або зум у широкому розумінні — міра наближення чи віддалення до об’єкта з метою краще його переглянути при використанні мобільних додатків.

User-friendly — дружній до користувача, інтуїтивно зрозумілий. Найчастіше вживається як характеристика інтерфейсу.

NavMesh, або навігаційна сітка — представлення території для навігації у вигляді сукупності опуклих багатокутників, що робиться для зручного пошуку шляху та навігації.

NavMesh Agent — фігура, що здійснює переміщення навігаційною сіткою.

Слайдер, або смужка прокручування - це графічний елемент інтерфейсу, за допомогою якого користувач може встановлювати значення, переміщуючи індикатор по горизонталі чи вертикалі (в залежності від положення самого слайдеру).

FOV (field of view), або поле зору — це відкрита зона спостереження, яку людина може бачити очима або через оптичний пристрій. Що стосується оптичних пристроїв та датчиків, FOV описує кут, через який пристрої можуть вловлювати електромагнітне випромінювання.

Кватерніони — чотирьохвимірне розширення множини комплексних чисел, тобто гіперкомплексні числа. Вони були запропоновані Вільямом Гамільтоном у 1943 році. Нині широко використовуються у комп’ютерних технологіях для опису обертання об’єктів.

First Person (camera), fp, fpc, (камера) від першої особи — це будь-яка графічна перспектива, відображена з точки зору персонажа гравця (термін запозичено з ігрової індустрії).

Pathfinding, або пошук шляху — це знаходження комп’ютерною системою найкоротшого шляху між двома точками. Pathfinding algorithms — алгоритми, що забезпечують знаходження такого шляху.

ВСТУП

Зараз складно уявити своє життя без різноманітних систем навігації. Якщо раніше найчастіше використовували карту і компас, щоб орієнтуватися на місцевості і діставатися до певних точок призначення, то зараз в більшості випадків у цьому немає потреби. Не потрібно навіть задумуватися, щоб привести найпростіший приклад: зараз майже у кожного в телефоні є система GPS, яка допомагає нам у подорожах, у прогулянках у незнайомому місті або навіть у своєму рідному, коли треба дістатися до якихось закладу чи будівлі, де ми раніше не бували.

Але такий метод не завжди дієвий, адже коли справа доходить до навігації будівлею, то GPS не працює. Потрібно знаходити якісь інші методи для здійснення навігації всередині, адже часто будівлі бувають величезними або з великою кількістю кімнат. Найпростіший приклад — корпус університету, де є багато аудиторій. І першокурсник, який, знаючи номер кабінету, при цьому не знає, як до нього дістатися.

Навігаційні системи широко використовуються у відкритому середовищі, але системи навігації в приміщеннях ще знаходяться на ранніх стадіях розвитку [1].

Саме тому розроблення такого застосунку є актуальною проблемою, яка вирішена у запропонованій роботі. Рзроблений модуль є частиною програмного продукту, який зможе допомагати людям швидше і зручніше здійснювати навігацію нашим навчальним корпусом. Основний акцент при розробці було здійснено на достатню точність роботи алгоритму навігації, його зручність у користуванні та дешевизну, адже передбачається, що додатком користуватиметься багато людей.

Міркування вище доводять актуальність розробленого продукту, адже подібна система суттєво полегшує навігацію в приміщенні, хоча рідко де використовується, тому такого роду продукт буде гарним нововведенням.

1 ЗАДАЧА РОЗРОБКИ ПІДСИСТЕМИ ПРОЕЦІЮВАННЯ ТРАЕКТОРІЇ РУХУ НА ОБ'ЄКТИ РЕАЛЬНОГО СВІТУ

Метою розробки є створення програмного модулю, що є підсистемою мобільного додатку навігації у закритому приміщенні (наприклад, корпус навчального закладу). Він дозволятиме здійснювати в ньому навігацію, а саме обирати бажану аудиторію або інше місце корпусу як кінцеву точку навігації, бачити весь маршрут від початку до кінця, обирати режим, в якому користувач хоче здійснювати перехід, а також забезпечуватиме користувача зручним інтерфейсом, що полегшить користування програмним продуктом.

Важливим пунктом у роботі програми має бути її зручність. Потрібно розробити програму так, щоб вона була максимально інтуїтивно простою, і користувачу було все зрозуміло. Наприклад, розробити інтерфейс так, щоб у користувача не виникало запитань, що робить та чи інша кнопка при її натисканні або обрання певного елементу у випадаючому списку. Крім того, слід пам'ятати, що у людей різні смартфони з різною роздільною здатністю екрану. До зручності можна віднести і можливість задання кінцевої точки призначення не тільки на видимій частині карти, тобто бажано забезпечити користувача можливістю перегляду карти не тільки там, де він знаходиться. Не слід і забувати про те, що доповнена реальність має однозначно і відразу бути зрозумілою користувачу.

Отже, програмний продукт повинен:

— давати можливість користувачу обирати кінцеву точку призначення;

— показувати шлях до неї;

— допомагати користувачу здійснювати прохід до бажаної точки призначення показом шляху за допомогою доповненої реальності;

— забезпечувати перегляд всієї карти, по якій здійснюється навігація;

— підлаштовуватися під різну роздільну здатність екрану;

— мати зрозумілий всім користувачам інтерфейс.

Враховуючи все вищесказане, можна побудувати діаграму прецедентів уже для всієї системи (вона також включатиме початкову синхронізацію) (рисунок 1.1).

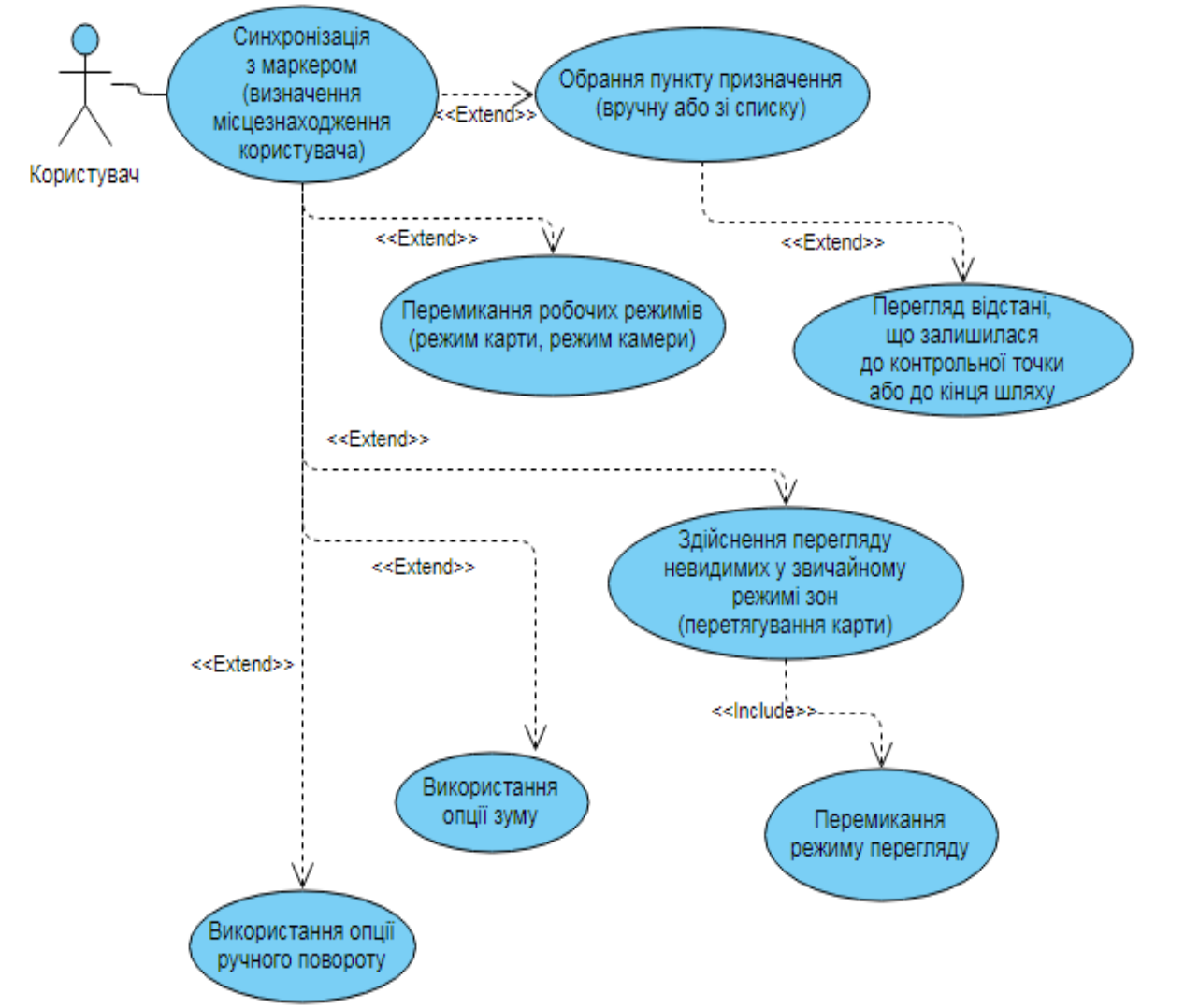

Рисунок 1.1 — Діаграма прецедентів системи

Для зручності не зайвим буде і додання режиму "зуму" (наближення і віддалення від карти), а також засобів реалізації перегляду маршруту на карті з різних сторін (поворот карти).

Для перегляду карти можна також зробити спеціальний режим, який дозволить рухати карту. Це забезпечить перегляд тих її частин, яких у звичайному режимі не видно.

Висновки до розділу 1

Отже, в даному розділі було сформульовано головну мету даної роботи, а саме розробку додатку для навігації всередині будівлі для смартфонів з використанням доповненої реальності. Також було продумано, якими характеристиками має володіти розроблюваний продукт, в тому числі підлаштування додатку під роздільну здатність пристроїв, а також user-friendly інтерфейс.

Також було розроблено і наведено діаграму прецедентів, що показує, які дії користувач може і має виконувати, користуючись розробленою системою.

2 АНАЛІЗ ПРОБЛЕМИ СТВОРЕННЯ ПІДСИСТЕМИ ПРОЕЦІЮВАННЯ ТРАЕКТОРІЇ РУХУ НА ОБ'ЄКТИ РЕАЛЬНОГО СВІТУ

Для створення зручного та раціонально побудованого додатку варто розглянути саме поняття доповненої реальності, та оцінити, як краще її використати для конкретного випадку. Також слід розглянути та проаналізувати існуючи аналоги.

2.1 Доповнена реальність та її використання

Augmented reality, або доповнена реальність — це доповнення фізичного світу додатковими даними (цифровими) з використанням пристроїв (наприклад, смартфонів) і програм [2]. Простіше кажучи, доповнена реальность — це поєднання цифрового світу з фізичним і, таким чином, збільшення досвіду від реального світу [3].

AR виділяється з рамок мобільних обчислень тим, що просторово і пізнавально долає межу між віртуальним та реальним світами. З AR, цифрова інформація стає частиною реального світу, принаймні у очах користувача [4].

У доповненій реальності віртуальні об'єкти проеціюються на реальне оточення.

Доповнена реальність [5] – це система, яка має наступні 3 характеристики:

— поєднує віртуальне і реальне;

— взаємодіє в реальному часі;

— працює в 3D.

Технологія AR була винайдена з 1950-х років. Доповнена реальність як інструмент для освіти та навчання вивчається з 1980-х та 1990-х років. У цьому контексті вона забезпечує безпечне середовище, в якому студенти та слухачі можуть взаємодіяти з віртуальними об'єктами та сценаріями, можливо, допускаючи помилки, не травмуючись чи завдаючи шкоди [6]. Системи з доповненою реальністю можуть покращувати виробничі процеси [7]. Мотивація, яка стоїть за технологією, — це

посилення візуального ефекту, який міг би допомогти користувачам легше припускати певний об'єкт. Тому інтерактивне 3D-зображення, що накладається на реальну сцену, може підвищити ефективність у навчанні, наприклад, підготовка хірургів для проведення операцій та ін. Доповнена реальність все частіше використовується для збільшення досвіду користувачів у різних завданнях.

Існує кілька проблем, пов’язаних з розробкою додатку навігації з доповненою реальністю.

Одна з них — це вибір інструментарію, за допомогою якого слід додавати об’єкти до реального світу. Суть полягає не в виборі інструменту як такого, а в тому, для яких платформ створений цей пакет. Відтак є 2 популярних аналоги: ARCore та ARKit. Перший з них націлений на девайси на базі android, коли останній — на iOS. Є і інші, навіть платні, як, наприклад, Wikitude.

Наступна проблема — це вибір методу, за допомогою якого слід додати об’єкт до реального світу. Наприклад, ARCore підтримує таку фундаментальну для доповненої реальності функцію, як виявлення площин (стіни, столи, підлога). Тому стоїть вибір, яким чином розміщати об’єкти. Можна це зробити, прив’язуючись до площини, а можна, просто оперуючи тільки координатами об’єктів, що будуть додаватися до реального світу. У кожного з цих методів є свої плюси і мінуси. Крім того, доцільність кожного з цих методів залежить і від об’єктів, які потрібно додати. Наприклад, використання виявлення площин доцільне тоді, коли в процесі роботи програми будуть використовуватися ці площини. Яскравим прикладом цього може слугувати гра "змійка" з використанням доповненої реальності, адже всі дії розгортаються на одній площині.

Ще одне питання, над яким потрібно задуматись, — як саме розмістити об’єкт чи об’єкти, щоб це було оптимально? Тут слід врахувати кілька факторів:

— скільки об’єктів потрібно розмістити? Тут слід знайти компроміс між тим, що не всі телефони надто потужні, і тим, що шлях користувачу має бути відразу зрозумілий;

— як саме в просторі їх розміщати? Можливо, їх слід розставити на контрольні точки шляху, вести лінію по підлозі або вказувати правильний шлях стрілкою.

Спробуємо розглянути конкретні випадки.

Спочатку проаналізуємо варіант, що передбачає використання стрілок, розташованих на підлозі, що вказують шлях (рисунок 2.1).

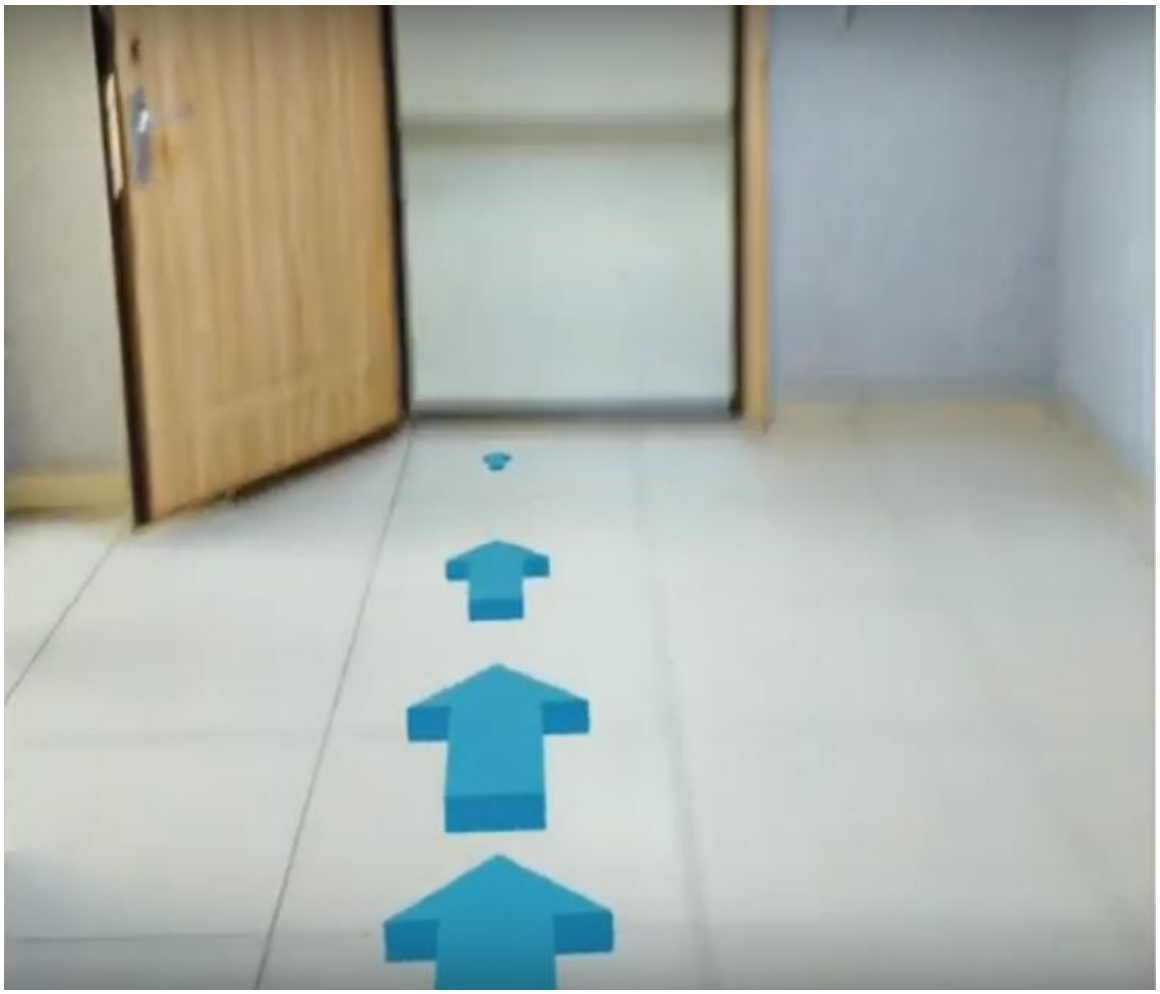

Рисунок 2.1 — Приклад додатку, де для показу обраного шляху використовуються стрілки

Така реалізація досить цікава, але має кілька проблем. По-перше, стрілки доповненої реальності мають бути розташовані на підлозі. Це означає, що потрібно підключити розпізнавання площин і розміщення на них об’єктів. Цей варіант не є найкращим, і ось чому. При вмиканні режиму з камерою найзручніше було б, якби

відразу показувався бажаний шлях. Але розпізнавання площин маже зайняти деякий час. Він може бути різний в залежності від деяких факторів, наприклад, освітлення або конкретного пристрою, будь то телефон чи планшет. Забігаючи наперед, слід додати, що пропонований додаток в ході розробки тестувався на різних телефонах. В залежності від їх потужності час розпізнавання площин був різним. З цим фактором неможливо боротися, тому треба мати його на увазі. В будь-якому випадку, користувач може натрапити на незручності, коли на екрані не буде показано шлях за допомогою доповненої реальності хай навіть 15 секунд, адже площини ще не розпізнано.

Додатковою проблемою є розпізнавання саме підлоги. Потрібно вводити додаткові алгоритми, що визначатимуть якусь із розпізнаних площин як підлогу, щоб потім на неї помістити віртуальні цифрові об’єкти. Так, можливо, підлога — це найнижча площина, але не можна бути впевненим, що це завжди буде так.

Звісно, можна розміщувати стрілки не обов’язково на підлозі. Розташування таких вказівників, наприклад, на півметра нижче самого пристрою, що використовується для навігації, може бути достойною альтернативою Проте різні люди мають різний зріст, і неможливо точно сказати, чи буде такий підхід досить зручним для кожного.

Слід зазначити, що особливістю доповненої реальності є те, що її об’єкти накладаються на зображення екрану на передній план, тобто зверху основного зображення [8]. Якщо додаток буде розроблено таким чином, аби він показував весь шлях від початку і до кінця, це не буде вірно. Аргументувати можна це тим, шо шлях, який показуватиме, де знаходиться бажане місце призначення, майже ніколи не буде прямим. Він матиме кілька або багато поворотів, можливо, навіть підйоми або спуски на інші поверхи. Саме тому він буде закриватися перешкодами, наприклад, стінами (рисунок 2.2). Якщо виникне ситуація, коли реальний шлях не видно повністю, а на екрані показано його від початку до кінця, це майже точно викличе незрозумілість або навіть обурення у користувача [9].

Останню проблему можна спробувати подолати декількома способами. Перший з них — використання додаткових алгоритмів, які будуть показувати тільки

видиму частину шляху, а все інше приховувати, поки користувач не пройде якусь його частину і не відкриє нові видимі ділянки.

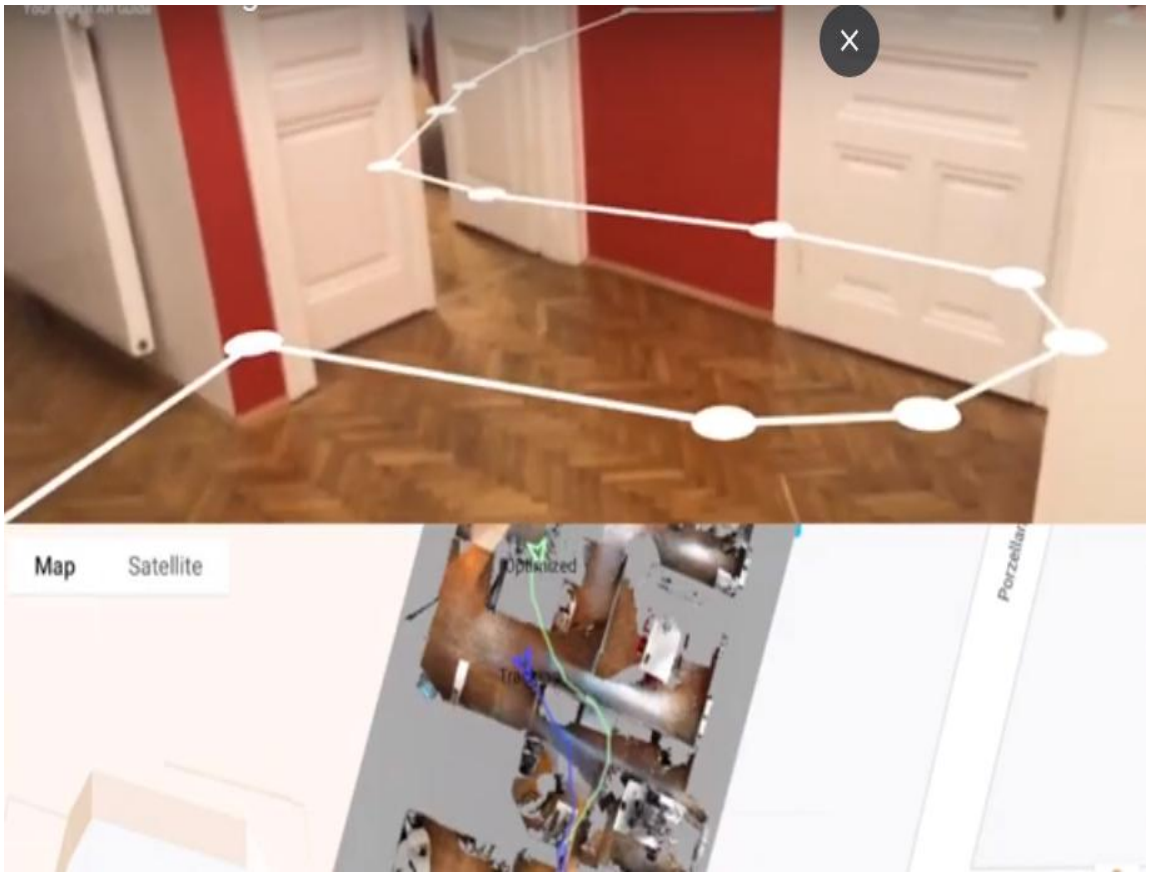

Рисунок 2.2 — Не достатньо коректне використання AR, адже лінія, що показує шлях, накладається поверх перешкод

Одним з таких алгоритмів є відомий в 3D моделюванні алгоритм трасування променів. Він справді використовується в деяких роботах по відображенню шляху за допомогою доповненої реальності [10]. Тут можна довго розмірковувати, чи воно варте того. Інший спосіб полягає в тому, щоб показувати шлях тільки до найближчої контрольної точки, вважаючи, що вона разом з користувачем буде належати одній прямій, що не перетинає перешкоди, і що шлях — це ломана лінія, вершини якої і є цими контрольними точками. Останній спосіб — це такий, як продемонстровано на рисунку вище, а саме відображення стрілок поступово по мірі того, як користувач просувається вперед, тобто створювати маршрут з ефектом згасання по радіусу. Це

рішення є менш ресурсоємним, і воно дає аналогічні, якщо не найкращі результати, ніж інші варіанти [11]. Можливо, цей спосіб є найкращим.

Такі ж проблеми можуть виникнути і при обранні методу показу шляху до бажаної точки призначення за допомогою лінії (рисунок 2.3), адже ці варіанти приблизно однакові по своїй суті.

Рисунок 2.3 — Приклад додатку, що вказує шлях за допомогою лінії доповненої реальності

У будь-якому випадку, підхід з використанням стрілок або лінії на підлозі так, щоб вони показували видиму частину шляху, мабуть, є найбільш зрозумілим для користувача, проте складним для реалізації.

Наступним є варіант використання стрілки, яка буде розташована не екрані начебто перед користувачем, але своїм поворотом завжди показуватиме, куди рухатися, щоб досягти найближчої контрольної точки. Звісно, у такому разі потрібно подбати про алгоритм, який би забезпечував постійний поворот стрілки так, аби вона з достатньою точністю показувала напрям до найближчої контрольної точки. Під достатньою точністю у даному випадку розуміється такий поворот, при якому

користувачу без труднощів було б зрозуміло, куди рухатися, щоб дістатися до точки, і в разі такого руху він справді б до неї дійшов.

Такий метод є менш використовуваним у системах навігації, проте він також знайшов свої місце у цій сфері. Одним з яскравих прикладів є використання схожого підходу, наприклад, у серії ігор Need For Speed (NFS), де для навігації містом гравцеві пропонується керуватися стрілкою, що вказує шлях до найближчого повороту (рисунок 2.4).

Рисунок 2.4 — Приклад використання стрілки, що своїм поворотом вказує шлях до найближчої контрольної точки, в комп’ютерній грі з серії Need For Speed

Останній варіант, на думку автора, є найдоцільнішим для використання у подібному додатку. По-перше, він не вимагає використання розпізнавання площин, що пришвидшує роботу програми. По-друге, він не вимагає використання алгоритмів для приховування невидимої частини шляху, що знаходиться за перешкодами. По-третє, він є досить зрозумілим, аби користувач розумів, як йому рухатися, щоб досягти бажаної точки призначення. Цей метод за бажанням можна трохи удосконалити, а саме розміщувати ще якийсь віртуальний об’єкт на місці контрольної точки, що додасть більшої зрозумілості для користувача.

Так як це додаток навігації, звісно, варто звернути увагу і на пошук шляху, а значить, і на алгоритми пошуку шляху. На сьогодні існує багато таких алгоритмів. Кожен з них має свої плюси та мінуси. Наприклад, алгоритм пошуку вглибину простий в реалізації, але швидкість його роботи мала, а також він не завжди знаходить оптимальний шлях. Тому такий алгоритм зазвичай не використовують для пошуку шляху в серйозних проектах.

Для подібних проектів найчастіше використовують евристичні алгоритми пошуку шляху. Вони використовують евристичну функцію, що включає в себе вартість досягнення бажаної точки та оцінку відстані від початкової вершини до кінцевої. Це допомагає їм швидше знайти розв’язок задачі.

Алгоритм A* ("A-Стар") часто використовується в таких проектах і добре себе зарекомендував для рішення подібних задач. Більше того, існують навіть спеціальні інструментарії, в які цей алгоритм вже зашитий всередину. Таким є, наприклад, Unity NavMesh. Його використання в проекті може бути оптимальним, адже це дозволить зосередитися на інших задачах.

Потрібно також не забувати про інтерфейс. Крім того, що він має бути максимально дружнім до користувача (user-friendly), його конкретні елементи можуть бути не потрібні в певний момент роботи програми. Наприклад, якщо до програми буде додано зум, що дозволятиме краще розглянути карту, якою здійснюється переміщення, то в режимі камери з доповненою реальністю така опція буде не потрібна, як і сама карта, тому відповідний елемент інтерфейсу бажано прибрати у цей момент [12]. Крім того, інтерфейс має бути розроблений таким чином, аби на різних телефонах він відображався коректно: потрібно, щоб його елементи були закріплені відносно сторін екрану, аби при різних роздільних здатностях інтерфейс не з’їжджав і виглядав гармонійно, при цьому не закриваючи основну зону перегляду карти чи зображення з камери.

Програми з доповненою реальністю (AR) вимагають доволі точного позиціювання. В іншому разі доповнена інформація може бути розміщена неправильно, що призведе до заплутаного для користувача інтерфейсу.

2.2 Порівняння існуючих систем

Звісно, користуючись мережею Інтернет, можна знайти аналогічні рішення, аналогічні додатки. Крупні компанії розробляють схожі програми для замовників або продають їх тим, хто готовий заплатити за подібний софт.

Порівнюючи їх з системою, що розроблюється, слід зробити поправку на те, що вона створюється кількома людьми, в той час як схожі програми довгий час розробляються цілими командами компаній.

Пропоную розглянути декілька знайдених автором програм.

Перша з них — внутрішня навігаційна система доповненої реальності Inplaces, розроблена Cologne Intelligence. Почати слід з того, що і Inplaces, і система, що розроблюються, використовують один і той же фреймворк для роботи з доповненою реальністю, а саме ARCore. Проте розробка компанії також може використовувати і ARKit, що збільшує кількість пристроїв, на яких вона може бути запущена і працювати, а отже, теоретично може привабити більше користувачів. Обидві програми не залежать від апаратних інфраструктур, наприклад, BLEB (Bluetooth low energy beacon). Важливим показником є точність. Команда розробників Inplaces стверджує, що їх програма працює з точністю позиціювання до 1 сантиметра. В цей час розроблювана за співучастю автора програма має точність близько одного метра, що є повністю задовільно для невеликих будівель, таких, як навчальний корпус. Хоча, звісно, для аеропортів такої точності буде трохи замало.

Обидві системи використовують доповнену реальність, проте у Inplaces її трохи більше (наприклад, іконки під час прибуття до точки призначення). У обох систем зручний сучасний інтерфейс.

Цікаво те, що Inplaces підтримує багатьох користувачів одночасно та вимагає їх авторизації, і це дозволяє діставатися не тільки до певних місць, а також і до інших людей.

В інтернеті також можна знайти декілька інших схожих систем позиціювання і навігації; інформації про них багато, проте не все можна дізнатися. Наприклад, є

Hubbell IPS та IndoorAtlas. Якщо IndoorAtlas має точність від 3 до одного метра, то про Hubbell IPS автор, на жаль, не знайшов такої інформації. Проте автори системи стверджують, що програма високоточна (hyper-accurate), тому можна лише припустити, що додаток має точність в кілька сантиметрів або трохи більше.

Ці системи орієнтовані на навігацію і позиціювання, вони використовують додаткові технології. Якщо детальніше, Hubbell IPS використовує Visible Light Communication та Bluetooth (можливо, це вимагає додаткового обладнання), а IndoorAtlas може використовувати Bluetooth beacons та Wi-Fi тріангуляцію для оптимізації.

Проте наведені вище додатки, розроблені командами професіоналів з різних компаній, не мають доповненої реальності. Звісно, це не обов’язкова частина, проте вона додає програмам певної жвавості і цікавості, що, скоріше за все, позитивно сприймається користувачами. Звідси слідує, що в цих додатках навряд чи використовується ARCore або ARKit.

Наведені міркування продемонстровані в таблиці 2.1.

Таблиця 2.1. Порівняння схожих систем

Характеристика	Розроблювана система	Inplaces	Hubbell IPS	IndoorAtlas
Доповнена реальність	+	+	-	-
Фреймворк для ДР	ARCore	ARCore, ARKit	-	-
Точність	Приблизно 1 метр	До 1 сантиметра	можливо, від кількох сантиметрів і трохи більше	1-3 метра з подальшою оптимізацією
Додаткове обладнання	Не потребує	Не потребує	Є ймовірність (Bluetooth)	Bluetooth beacons, Wi-Fi

Цікавим доповненням розроблюваної системи є те, що вона має так званий режим перегляду. Користуючись ним, власник пристрою може переглядати карту так, як йому захочеться: перетягувати її в бажану сторону, наближати її чи повертати.

Проаналізувавши вищенаведену таблицю, можна стверджувати, що розроблювана програма матиме найнижчу вартість при досить непоганій точності.

Висновки до розділу 2

Отже, в даному розділі було розглянуто поняття augmented reality, або доповнена реальність, що означає додання до зображення реального світу цифрових даних за допомогою смартфону. Було наведено основні характеристики AR, а також розглянуто його історичну складову.

Було розглянуто проблеми, пов’язані з використанням доповненої реальності у розроблюваному додатку, наприклад, вибір фреймворку. Окремо було приділено уваги підходам, за допомогою яких можна показувати шлях у мобільному додатку навігації всередині приміщення, їх плюсам і мінусам, а також труднощам, що з ними пов’язані. На основі цього аналізу було обрано AR фреймворк ARCore, а підхід відображення — стрілка перед екраном девайсу користувача, що своїм поворотом показує напрямок руху. В кінці розділу було проведено порівняльний аналіз розробленої системи зі схожими, на основі якого можна стверджувати, що вона має деякі ключові переваги над ними.

3 ОПИС ПОБУДОВАНОЇ МОДЕЛІ ТА ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

Модуль системи був розроблений з використанням Unity та мови програмування C#. Відтак, в проекті ми маємо сцену з об’єктами, до яких прикріплені скрипти та інші компоненти.

Пропоную розглянути деякі ключові моменти створення програмного модулю.

3.1 Опис програмної реалізації

Спочатку слід уточнити, яка саме частина системи розробляється для всього мобільного додатку (рисунок. 3.1)

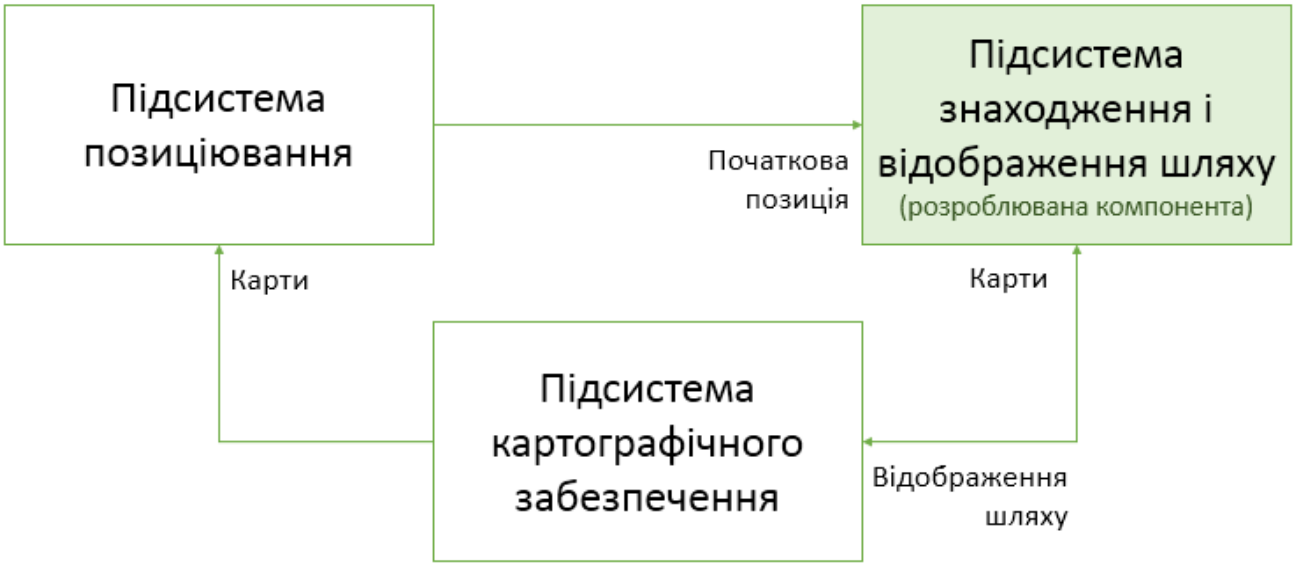

Рисунок 3.1 — Взаємодія компонентів системи

Як видно з рисунка вище, розроблюваною програмою є підсистема знаходження і відображення шляху, при цьому для її нормального функціонування потрібні дані з двох інших компонент: початкова позиція користувача з підсистеми позиціювання і карти з підсистеми картографічного забезпечення.

Говорячи про об’єкти на сцені, візьмемо, наприклад, SpherePointer або PointerController. Першого, наприклад, добре видно при роботі програми (сфера), тоді як другий створений як пустий об’єкт тільки для того, щоб задати частину поведінки програми. Всі об’єкти з компонентами є дочірніми сцені, яка має назву FloorNavigation; це можна побачити в лівій верхній частині екрану при запуску проекту (рисунок 3.2). Це і складає основу архітектури системи.

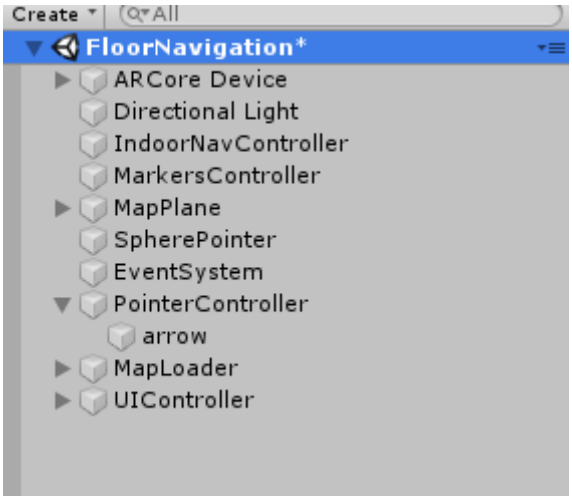

Рисунок 3.2 — Всі об’єкти є дочірніми до сцени FloorNavigation

Другим важливим пунктом у архітектурі є прикріплення різноманітних компонентів до об’єктів. Це є одним з основних принципів Unity. Найчастіше цими компонентами є так звані скрипти — файли з кодом, написаним на мові C# для визначення поведінки об’єктів, проте можуть бути і інші компоненти. Наприклад, до сфери, яка показує положення користувача в системі навігації, було прикріплено компонент LineRenderer, який потім в коді було запрограмовано так, щоб він показував шлях до точки призначення.

Для знаходження шляху було використано вбудований в Unity компонент навігації. (рисунок 3.3).

Суть полягає в тому, що до карти, по якій планується здійснювати навігацію (в даному проекті це об’єкт MapPlane), додається компонент Nav Mesh Surface.

Після цього можна задати деякі параметри навігації (наприклад, відстань, на яку об’єкт, що здійснює рух, може підходити до стін і т.д.) (рисунок 3.4).

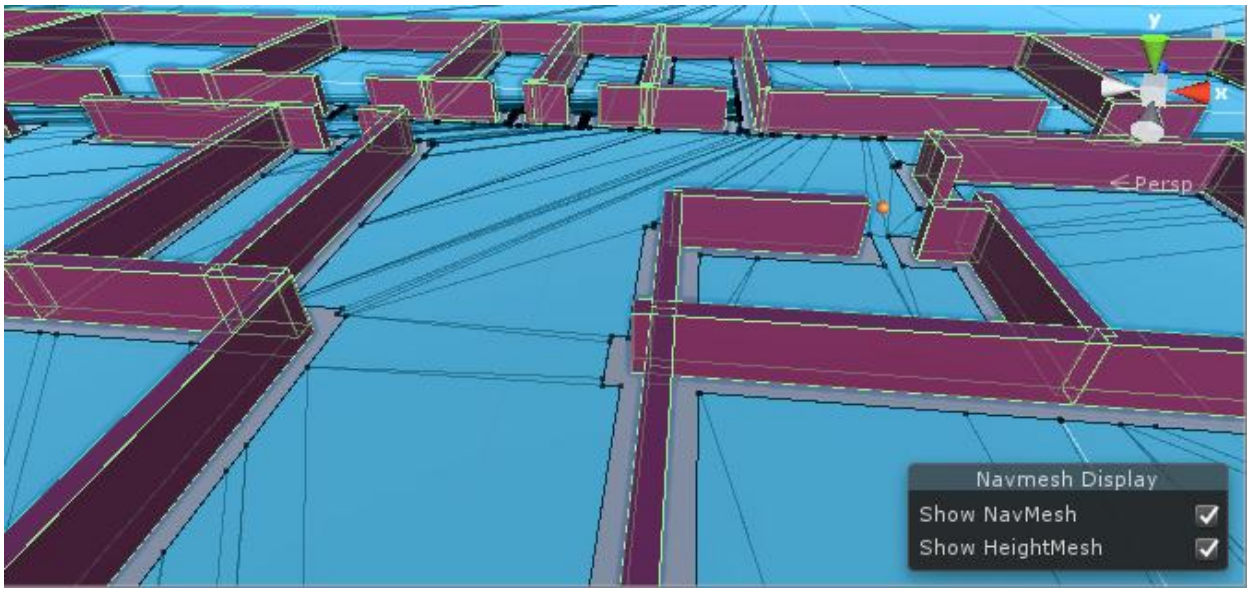

Рисунок 3.3 — NavMesh, або навігаційна сітка

Після цього необхідно "запікти" (bake) навігаційну сітку, що будується на основі обраної карти. Тут слід зазначити, що карта повинна мати стіни, також побудовані як об’єкти, інакше навігація не працюватиме, адже навігаційна сітка просто не побудується (об’єкт MapPlane має дочірні об’єкти Obstacle).

До об’єкту, який буде здійснювати навігацію (SpherePointer), необхідно додати компонент Nav Mesh Agent. Його можна також налаштувати (наприклад, задати швидкість об’єкту, що рухається) (рисунок 3.5).

Для задання кінцевого місця призначення вручну було обрано довге натиснення на карту в якусь точку. Це зменшує шанс випадкового задання не потрібної точки призначення звичайним натисненням.

Для розрізнення взаємодії з елементами інтерфейсу і картою до останньої було додано скрипт Map Plane Click Handler. Завдяки йому при натисненні на елемент інтерфейсу система не сприймає це як взаємодію з картою. Саме в ньому реалізовано алгоритм довгого натиснення (long click). Запам’ятовується час натиснення на карту та знаття пальця з екрану; оперуючи цими величинами, можна встановити потрібний час спрацьовування задання точки призначення. Час для лонгкліку було обрано оптимальний — 1 секунда.

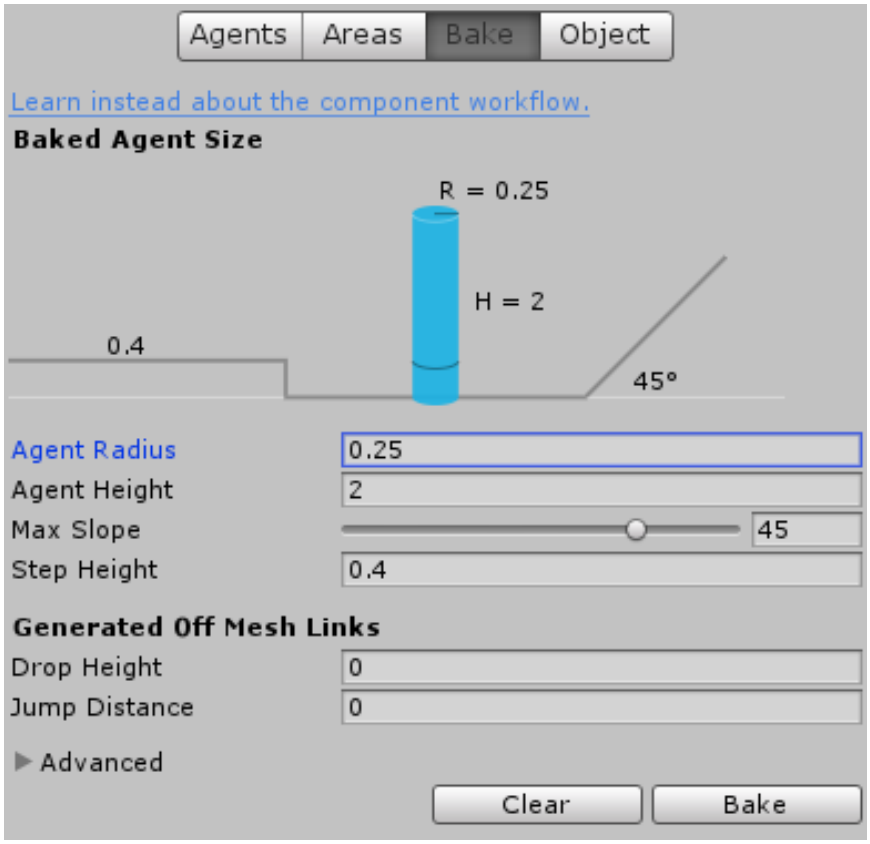

Рисунок 3.4 — Одна з вкладок налаштування компоненту навігації, в тому числі і мінімальної відстані від NavMesh агента до перешкод (стін)

Якщо довге натиснення здійснено, викликається метод саме зі скрипту UISrcipt, який називається OnMapClick, що отримує як аргумент точку призначення. Вже в ньому через Nav Mesh Agent задається кінцеве місце призначення. Оскільки можливий і інший варіант вибору кінцевої точки, а саме через випадаючий список, тому в цьому методі також встановлюється значення цього списку в "не обрано". Також тут викликається метод DrawPath, шо і відповідає за відмальовування шляху на карті.

Unity NavMesh Agent за замовчуванням розроблено так, що як тільки обрана кінцева точка призначення, то агент відразу починає свій рух. Враховуючи, що у даній системі агент, що здійснює рух, прив’язаний до користувача, така його поведінка буде неправильною, адже сфера-вказівник має показувати своїми рухами реальні рухи користувача. Саме тому у описаному вище методі також є рядок, що відміняє таку

дію, а саме негайний рух після обрання пункту призначення. Це робиться за допомогою задання поля isStoped об’єкту агента у значення false.

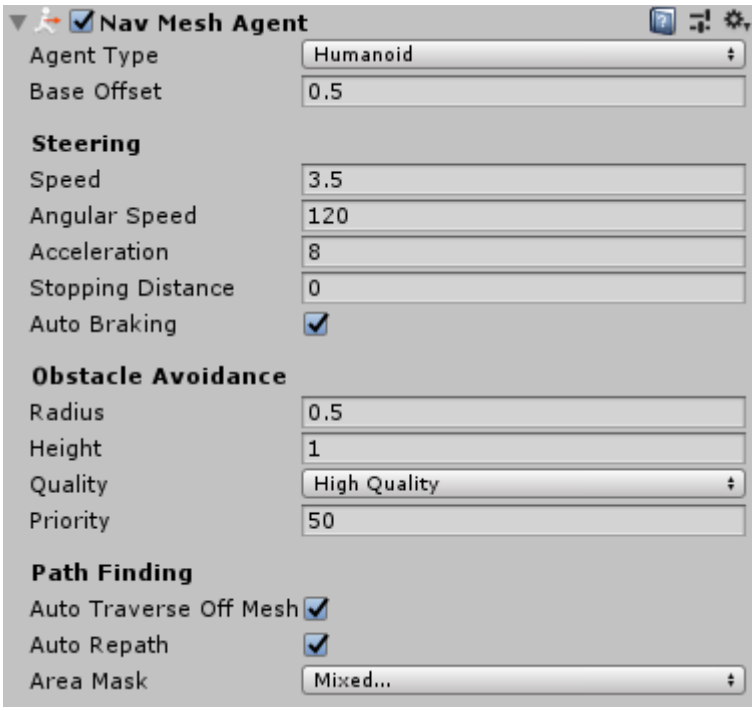

Рисунок 3.5 — Налаштування NavMesh Agent (швидкість агента, радіус обходження перешкож і т.д.)

Розглянемо роботу методу DrawPath. Як вже було сказано, для відображення шляху використано компонент LineRenderer, приєднаний до агента. Спочатку цьому компоненту задається кількість вершин, які буде мати наш шлях, представлений у вигляді ломаної лінії. Її ми можемо дізнатися з властивості path.corners агента після того, як кінцеву точку призначення вже обрано. Після цього в якості першої вершини встановлюється позиція сфери-вказівника, після чого компоненту задаються і інші вершини з властивості path.corners.

Робота програми при виборі точки призначення, що описана, добре видна на рисунку 3.6.

Окремо слід розглянути і випадок, коли кінцева точка призначення обирається не вручну, а з випадаючого списку. На екрані він розташований справа зверху.

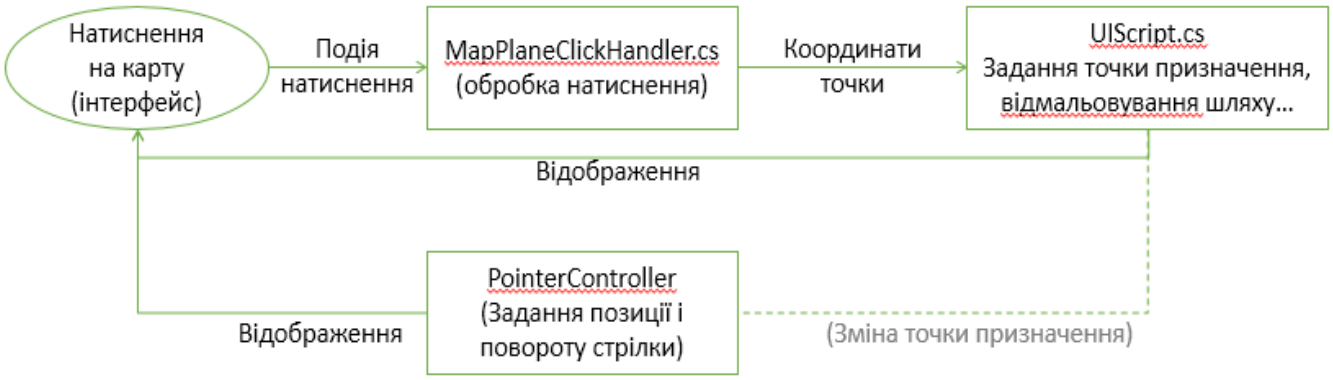

Рисунок 3.6 — Робота програми на прикладі обрання кінцевої точки призначення.

Коли ми кажемо випадаючий список, то маємо на увазі елемент інтерфейсу, при натисненні на який з'являється якась кількість опцій, що можуть бути обрані користувачем. Проте уважно поглянувши на вікно властивостей випадаючого списку, можна помітити, що спочатку цей список пустий (рисунок 3.7), проте під час роботи програми варіанти є.

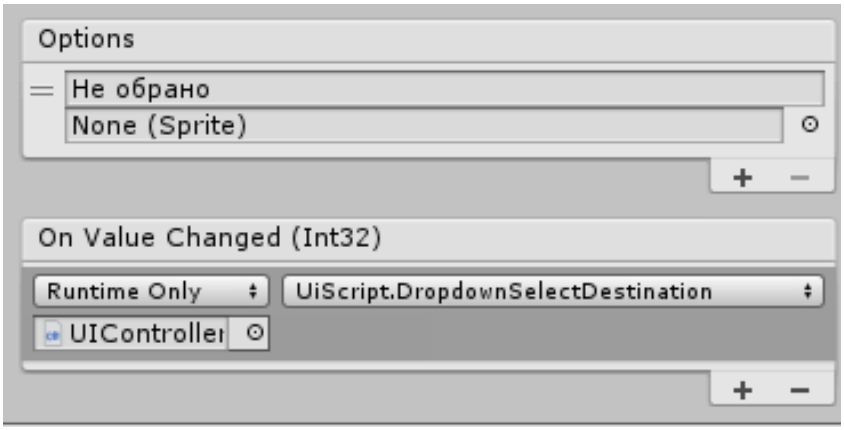

Рисунок 3.7 — Частина вікна властивостей випадаючого списку, що відповідає за обрання кінцевої точки призначення.

Це пов'язано з тим, що список заповнюється динамічно, коли програма запускається. Це виконується іншим модулем, який в межах даної дипломної роботи

не розроблявся, з використанням метаданих, тому не будемо зупинятися на цьому моменті детальніше.

На цьому ж рисунку 3.7 показано, що подія обрання одного з варіантів випадаючого списку обробляється в UIScript-і, а саме методом DropdownSelectDestination.

Опишемо роботу цього методу. Спочатку перевіряється, чи не було натиснуто варіант "Не обрано". Якщо це так, тоді в об’єкті NavMesh Agent викликається метод ResetPath, який скидає встановлену поточку кінцеву точку і не встановлює нову. В іншому випадку, в тимчасову змінну destinationPoint записуєтья кінцева точка, що відповідає координатам обраної у випадаючому списку аудиторії. Після цього перевіряється, чи знаходиться ця аудиторія на поточному поверсі. В залежності від цього в кінці методу в NavMesh Agent через SetDestination встановлюється або кінцева точка призначення, що була раніше записана в тимчасову змінну, або точка, що є координатами найближчих сходів, якщо треба піднятися або опуститися на інший поверх, попередньо також записана в destinationPoint. В останньому випадку обраховується оптимальний шлях спочатку до сходів, а після переходу на інший поверх користувач має змогу продовжити навігацію до бажаної аудиторії.

Одним з найголовніших файлів, які було написано в ході розробки модулю, є UIScript, до прикріплений до UIController. Він взаємодіє зі всіма елементами інтерфейсу програми, наприклад, з перемикачем між режимами показу карти і камери. Це перемикач ShowMapToggle, і одним з його обробників подій є метод ToggleMap, який і вмикає або вимикає потрібні компоненти. Наприклад, в режимі карти сама карта відображається, а стрілка, що веде в потрібному напрямі — ні, а в режимі камери навпаки.

Якщо розглянути цей момент детальніше, то можна помітити, що подія перемикання режимів роботи програми підхоплюється двома обробниками події в двох різних класах. Це добре видно у налаштуваннях цього перемикача (рисунок 3.8).

Першим обробником якраз і є метод ToggleMap в класі UIController. По суті, для перемикання режимів треба в залежності від значення перемикача вимкнути одні об’єкти і компоненти, а інші увімкнути.

Так, наприклад, якщо увімкнений режим камери з доповненою реальністю, то це означає, що метод вимкне карту.

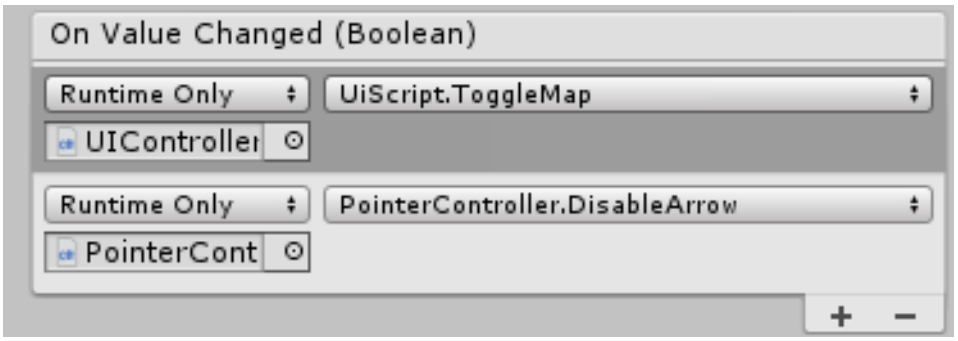

Рисунок 3.8 — Обробники події перемикання режимів роботи програми

Згадуючи, що карта в даному проекті має ще і перешкоди (стіни), представлені у вигляді дочірніх об’єктів, зрозуміло, що потрібно в циклі отримати в них компоненти типу Renderer і відключити їх. Крім того, відключити також потрібно і вказівник місцезнаходження користувача (аналогічним чином через встановлення в компоненті типу Renderer поля enabled в false), і компонент LineRenderer, що показує шлях на карті. Також потрібно відключити деякі елементи інтерфейсу, наприклад, слайдери для зміни зуму і повороту (їх буде описано далі). Натомість необхідно включити відображення стрілки доповненої реальності, яка і має показувати шлях в режимі камери.

Зрозуміло, що при перемиканні назад у режим карти метод ToggleMap зробить все навпаки, тобто присвоїть елементам, згаданим вище, протилежні значення.

Незважаючи на режим, в якому здійснюється робота, слід пам’ятати, що кінцева точка призначення може бути задана або не задана. В другому випадку при перемиканні в режим з доповненою реальністю правильним і логічним буде не відображення стрілки. Саме для цього і потрібен ще один обробник події перемикання режимів — метод DisableArrow у класі PointerController.

Для зручності користувача було додано також деякі додаткові опції.

Точно здійснювана навігація — це добре, але така програма стане більш цінною, якщо можна буде переглядати карту в різних конфігураціях наближення, повороту та місця, що переглядається в певний час.

Спершу пропонується розглянути зум (наближення).

Зазвичай при розробці подібного функціоналу для наближення і віддалення використовують таку властивість камери, як поле зору (FOV, field of view). Це величина, яка є кутом, зазвичай вимірюється в градусах і дорівнює значенню між крайньою лівою видимою межею і правою.

Як вже зазначалося, при розробці програми використовувався AR Framework ARCore. Однією з його особливостей є те, що поле зору певним чином пов’язане з його функціональністю, і його зміна може привести до певних збоїв у роботі цього інструменту. Саме тому він не дозволяє користувачу змінювати FOV.

В такому випадку для забезпечення зуму було обрано іншу стратегію. При взаємодії з інтерфейсом програма змінює не field of view, а відстань від камери до MapPlane, тобто до об’єкта з текстурою карти.

Розглянемо детальніше цю функцію модулю.

З правої сторони екрану розташований один з елементів інтерфейсу, який є слайдером, який так і підписаний — "зум". За замовчуванням він встановлений на мінімальне значення — 5. Це означає, що з самого початку роботи програми камера розташована максимально близько до карти.

Максимальне та мінімальне значення, яке може мати слайдер, також можна змінювати. Це робиться за допомогою налаштувань слайдеру (рисунок 3.9). Хоча змінювати їх може розробник, звісно. Саме тому при заданні цих значень можна орієнтуватися, наприклад, на загальний розмір карти або на вподобання замовника.

Тепер перейдемо до того, як організована робота цього слайдера, тобто як саме він працює.

Коли користувач змінює стан слайдеру, перетягуючи його інтерактивну частину, генерується подія. Ця подія підхоплюється обробником, який встановлює розробник. Для цього потрібно обрати підходящий клас, написати в ньому метод, який певним чином реагуватиме на зміну стану слайдеру. Після цього слід додати цей

клас в спеціальне місце в налаштуваннях цього елементу інтерфейсу. Далі потрібно обрати попередньо написаний метод.

В цьому конкретному випадку зміна значення слайдеру, що відповідає за зум, оброблюється класом UIController (як і більшість елементів інтерфейсу). Функція, яка спрацьовує при цьому, називається OnSliderCameraZoomChanged (рисунок 3.10).

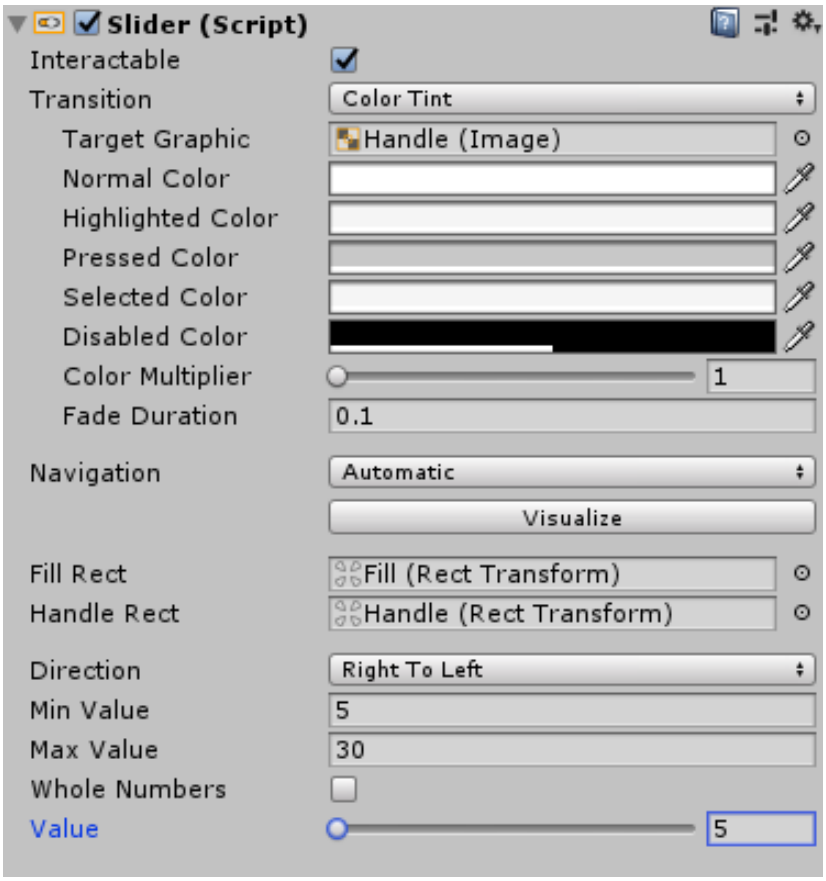

Рисунок 3.9 — Налаштування слайдеру в редакторі (внизу — задання максимального, мінімального значення та значення по замовчуванню)

Цей метод невеликий. Основна його задача — отримати компонент, прикріплений до камери, який називається FollowTarget, і поміняти в ньому значення змінної heightOverTarget. Після цього до об’єкту камери застосовується висота, що дорівнює сумі висоті вказівника, що показує місцезнаходження користувача (SpherePointer), і значення змінної heightOverTarget. І тільки після цього користувач бачить карту з новим значенням зуму.

Тут слід окремо сказати кілька слів про FollowTarget. Логічно, що під час переміщення користувача рухається і вказівник, який показує його місцезнаходження. Цей скрипт було спочатку додано для того, аби під час руху цього вказівника камера рухалася за ним, адже це зручно. Але потім для зручності до нього було додано і ці додаткові функції.

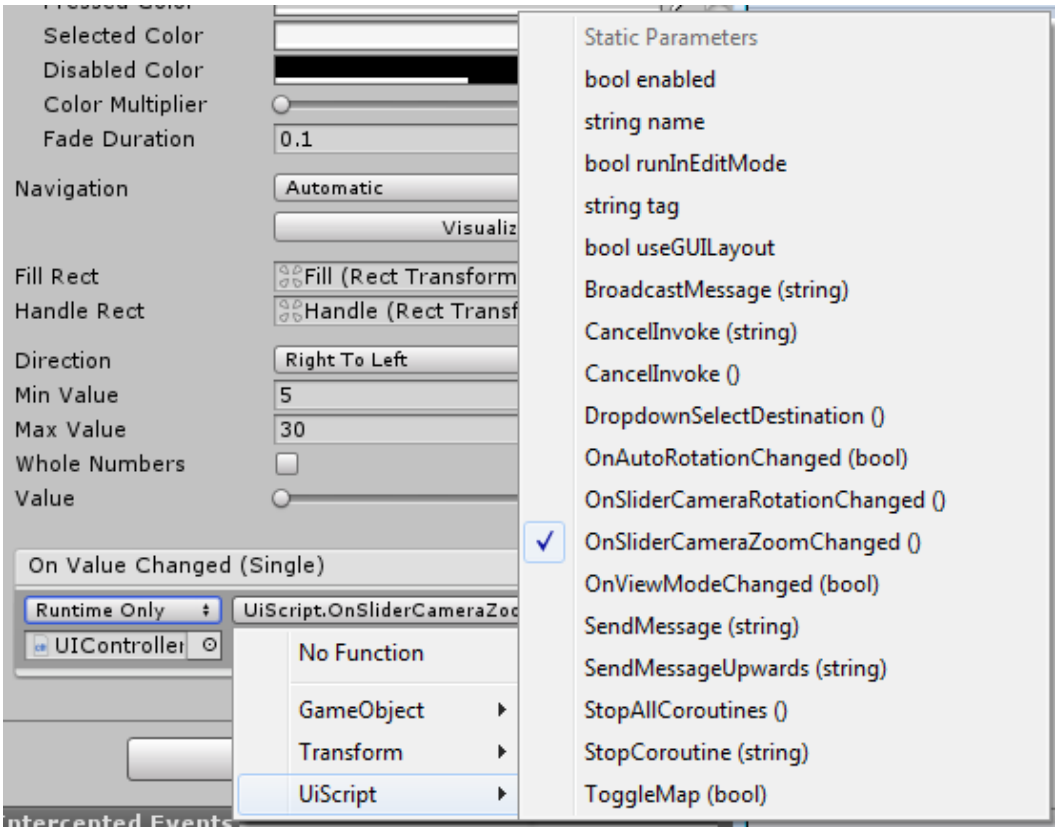

Рисунок 3.10 — Встановлення обробника події зміну стану слайдеру.

Наступною з них є поворот камери. Розглянемо її детальніше.

З самого початку, коли користувач запускає програму (або після синхронізації з маркером), значення повороту задається автоматично. За замовчуванням камера розташовується трохи позаду за SpherePointer і "дивиться" на нього; при повороті смартфону камера повертається так, щоб залишатися в тому ж положенні відносно сфери-вказівника. Це зручно, адже при використанні додатку камера наче йде за вказівником положення користувача.

Але якщо користувач, припустимо, захоче зупинитися і оглянути своє місцезнаходження з різних сторін для кращого розуміння ситуації, розроблений додаток надає йому таку можливість.

Для цього було також розроблено слайдер повороту, що розміщений в правій нижній частині екрану. За замовчуванням камера повертається автоматично, і тому слайдер спочатку заблокований для користувача. Для того, аби мати змогу користуватися їм і повертати камеру власноруч, потрібно скористатися спеціальним перемикачем, що розташований над слайдером. Він має назву "авто", і спочатку там стоїть галка. При її знятті слайдер стає інтерактивним. До цього перемикача, як і раніше, прив’язаний метод класу UIController, в даному випадку це OnAutoRotationChanged (рисунок 3.11).

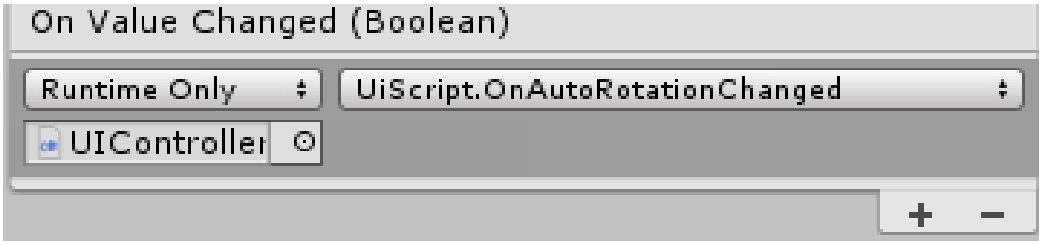

Рисунок 3.11 — Обробник події зміни стану перемикача "авто"

Цей метод є простим і невеликим. З самого початку він задає поле interactable слайду повороту відповідним чином (якщо галка стоїть — значення false, і навпаки). Далі йде перевірка: якщо перемикач було встановлено в значення автоматичного повороту, то початковий кут повороту встановлюється в 0.

В цьому методі також встановлюється значення AutoRotation об’єкта типу CameraState. Цей тип відповідає за поточний стан камери і був розроблений в іншому модулі для правильної роботи системи після синхронізації з маркером. Саме тому не буде розглянуто детально його роботу, хоча його методи використовуються в роботі поточної підсистеми.

Налаштування слайдера в цьому випадку аналогічні попередньому. Відрізняються тільки мінімальна та максимальна межа: логічно, що для повороту треба вказати значення градусів від 0 до 359 (рисунок 3.12).

Як і інший слайдер, цей також має свій обробник події зміни значення. Це метод OnSliderCameraRotationChanged, що, як і раніше, розташований у UIController-і (рисунок 3.13).

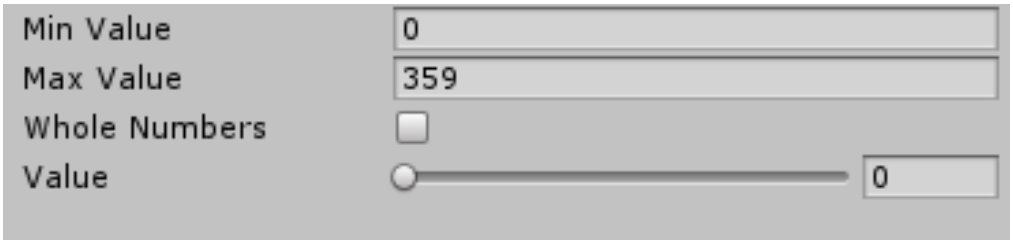

Рисунок 3.12 — Крайні значення слайдеру повороту — 0 і 359 градусів

В цьому методі створюється об’єкт типу кватерніон. За допомогою методу AngleAxis цього ж класу можна задати значення цього об’єкту, використовуючи значення градусів, яке і отримується від слайдеру. Після цього значенню поля Rotation об’єкту типу CameraState задається обрахований поворот.

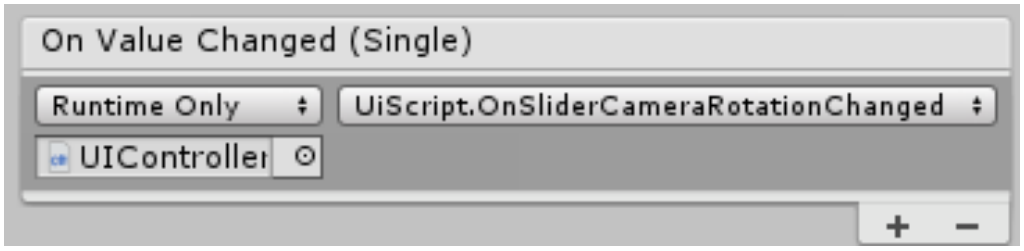

Рисунок 3.13 — Обробник події зміни значення слайдеру повороту

Подальші дії по заданню повороту камери відбуваються в класі FollowTarget. В ньому ми отримуємо значення повороту з об’єкту CameraState, використовуючи метод GetCameraRotation, і після нечисленних перетворень задаємо камері бажаний поворот.

Далі слід розглянути режим перегляду, що дозволяє перетягувати карту і дивитися її різні місця, які з самого початку роботи програми не видно. Для того, аби перейти в цей режим, до інтерфейсу було додано ще один перемикач. Він розміщений в правій верхній частині екрану, відразу під випадаючим списком.

З самого початку цей перемикач вимкнений, що означає, що робота здійснюється в звичайному режимі. Але якщо поставити галку, то додаток перемкнеться в режим перегляду.

По суті, цей режим особливо не відрізняється роботою від звичайного. Його відмінність полягає в тому, що тепер камера не рухається за вказівником місця, де знаходиться користувач, а також в тому, що тепер можна переглядати різні куточки карти.

Розглянемо детальніше, як він працює.

Для вже вищезгаданого перемикача було розроблено метод OnViewModeChanged в класі UIController (рисунок 3.14).

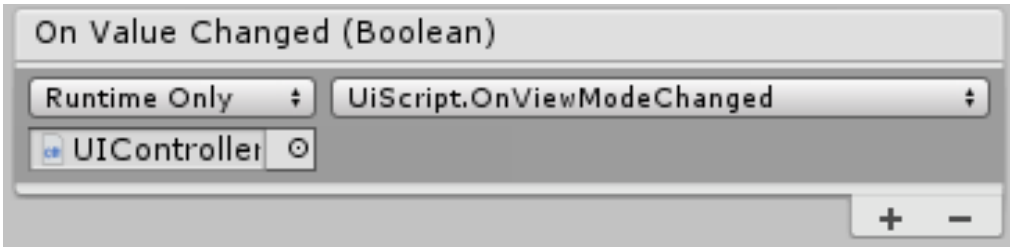

Рисунок 3.14 — Обробник події для перемикача режиму перегляду

Цей метод встановлює відповідне значення локальної змінної viewMode, а також значення змінної viewMode в уже знайомому скрипі FollowTarget, що прикріплений до камери.

Далі користувач має перетягнути карту, використовуючи дотик до екрану. Об’єкт MapPlane, що по суті є картою, має прикріплений до нього компонент під назвою MapPlaneClickHandler. Він і відповідає за взаємодію з картою (вже згадане вище встановлення контрольної точки довгим натисканням).

Цей клас, реалізовуючи декілька інтерфейсів, має методи OnBeginDrag і OnDrag. Вони і є відповідальні за роботу так званого drag&drop. Слід зауважити, що перший з них реалізовано, але його тіло пусте. Справа в тому, що для коректної роботи перетягування цей метод має бути обов’язково реалізований. А от останній передає різницю між поточною і попередньою позицією натискання в UIScript,

використовуючи метод SetDelta. Це об’єкт delta класу PointerEventData, який зберігає дані про подію.

В методі SetDelta спочатку відбувається перевірка, чи справді додаток було перемкнуто в режим перегляду (локальна змінна viewMode, що була згадана раніше). Якщо так, то перше, що необхідно зробити, підлаштувати перетягування так, щоб на різних пристроях воно працювало приблизно однаково. Це пов’язано з тим, що різні девайси мають різні розширення екрану, тому на одних пристроях перетягування може бути прийнятним по швидкості, а для інших — занадто повільне або швидке.

Для цього спочатку слід знайти максимальний розмір екрану, виходячи з розширення (у пікселях). Це продемонстровано у формулі (3.1)

$$screenSize = Max(screenWidth, screeHeight), \quad (3.1)$$

де $screenSize$ — шукана величина (максимальне розширення екрану в пікселях);

$screenWidth$ — розширення екрану в ширину;

$screeHeight$ — розширення екрану в висоту.

Після цього слід обрахувати величину, на яку буде змінюватися положення карти (3.2).

$$deltaVec3 = newDelta * sensitivity \ / \ screenSize, \quad (3.2)$$

де $deltaVec3$ — значення, на яке буде переміщено карту;

$newDelta$ — значення, що було отримане з MapPlaneClickHandler (різниця між поточною і попередньою позицією натискання);

$sensitivity$ — чутливість;

$screenSize$ — максимальне розширення екрану в пікселях по горизонталі чи вертикалі (знайдене в попередній формулі).

Слід додати кілька слів про чутливість. Чим більше це значення, тим більш різко буде зсовуватися карта; обране значення конкретно для цього проекту — 25, але його можна змінити, зсилаючись на побажання замовника.

Коли це значення розраховано, воно і використовується для зсуву карти. Хоча насправді на практиці відбувається зсув не карти, а камери, хоча для користувача це не грає ніякої ролі.

Крім того, в цьому методі відбувається перевірка, чи не вийшов користувач при перетягуванні карти за її межі. Якщо вийшов, то положення камери встановлюється таким, яким є положення відповідного краю карти (звісно, по висоті камера знаходиться вище над картою).

В FollowTarget змінна viewMode також використовується. В кінці файлу в залежності від її значення виконуються різні дії. І в тому, і в іншому випадку (їх 2, бо це змінна типу bool) відбувається задання повороту (відповідно до автоповороту чи вказаного користувачем) і наближення (точніше, висоти камери, заданий користувачем зум). Але у випадку звичайного режиму камера встановлюється приблизно над сферою-вказівником положення користувача, а при включеному режимі перегляду дорівнює значенню, порахованому за допомогою значення зсуву, розрахованому у попередній формулі.

В режимі камери було вирішено завжди показувати стрілку перед користувачем (точніше, перед камерою). Один з альтернативних варіантів — поміщати об’єкти, що направляють користувача, на контрольні точки шляху, таким чином ведучи до кінцевого місця призначення. Проте такий варіант було відкинуто: якщо користувач увімкне режим камери, а найближча контрольна точка буде позаду нього, то він з самого початку не буде розуміти, куди йти. В нашому ж випадку, стрілка завжди буде з’являтися перед ним, вказуючи правильний шлях, що не викличе у користувача складнощів.

На прикладі об’єкту і скрипта PointerController можна побачити, яким чином передаються аргументи з редактора Unity в C#-код (рисунок 3.15). Після такої операції ці об’єкти будуть доступні в класі PointerController при їх оголошенні.

В цьому ж класі розміщена логіка роботи цього вказівника (стрілки) Спочатку перевіряється, чи потрібно відображати стрілку (чи заданий шлях, чи показується карта). Якщо потрібно, то на основі позицій камери та вектору, що вказує прямо від камери, знаходиться позиція стрілки.

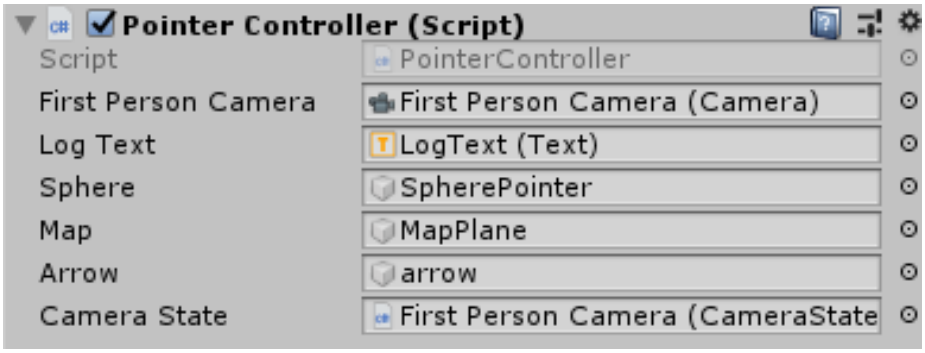

Рисунок 3.15 — Передавання аргументів в код

Саме позиція і відповідає за впровадження стрілки наче б то у реальний світ, що і є доповненою реальністю. Після цього визначається поворот стрілки, використовуючи кут (відносно осі y) між вектором від сфери, що вказує на місцеположення користувача, до найближчої контрольної точки та поворотом системи координат цієї ж сфери відносно позиції пристрою, що використовується, помноженим на одиничний вектор по осі z. Цей поворот системи координат потрібен для коректної роботи маркерів системи (інший модуль), тому без нього не обійтись. Позиція та поворот стрілки оновлюються кожен кадр, тому стрілка завжди показує правильний напрямок.

Далі пропонується розглянути більш детально встановлення позиції та повороту стрілки доповненої реальності.

При перемиканні в режим доповненої реальності стрілка знаходиться перед користувачем на певній попередньо встановленій відстані. Ця відстань була взята за 3 unity-одиниці (співпадають з метрами). В такому випадку вектор розміщення стрілки буде дорівнювати сумі двох векторів: вектор розміщення fpc (first person camera, камера від першої особи) та вектор, що показує напрям, куди спрямований погляд камери, помножений на відстань до стрілки (3.3).

$$\overline{ArrPos} = \overline{fpc.forward} * distanceToArrow + \overline{fpc.position}, \quad (3.3)$$

де $\overline{ArrPos}$ — позиція стрілки, яку потрібно розрахувати;

$\overline{fpc.forward}$ — вектор, що вказує напрям, куди "дивиться" камера;

$distanceToArrow$ — відстань від камери до стрілки, взята за 3;

$\overline{fpc.position}$ — позиція камери.

Це продемонстровано на рисунку 3.16. Для спрощення покажемо це в 2D-просторі, ігноруючи вісь y.

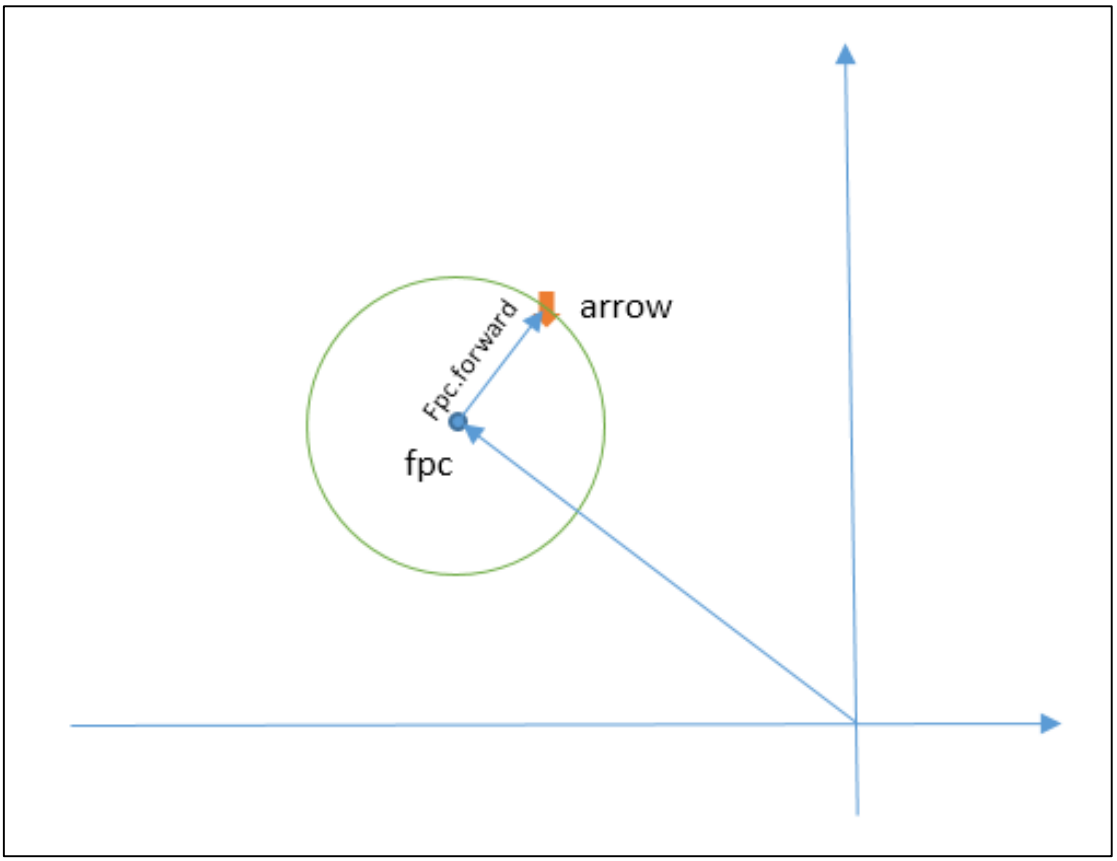

Рисунок 3.16 — Розміщення стрілки доповненої реальності

Задати стрілці правильний поворот — не менш важливо, ніж позицію. Цей поворот і буде відповідати за напрямок, куди повинен рухатися користувач, тому його розрахунок повинен бути виконаний правильно, аби людина, що здійснює навігацію, не заплуталася.

Для того, аби знайти правильний поворот, спочатку треба порахувати вектор, що сполучає користувача (spherePointer) та точку, куди потрібно дістатися (destination)) (3.4). Відповідно до розрахунку оновимо попередній рисунок 3.16 до 3.17.

$$\overline{destVector} = \overline{destination} - \overline{spherePointer.position}, \quad (3.4)$$

де $\overline{destVector}$ — шуканий вектор;

$\overline{destination}$ — вектор, що вказує на найближчу контрольну точку;

$spherePointer.position$ — позиція користувача (spherePointer).

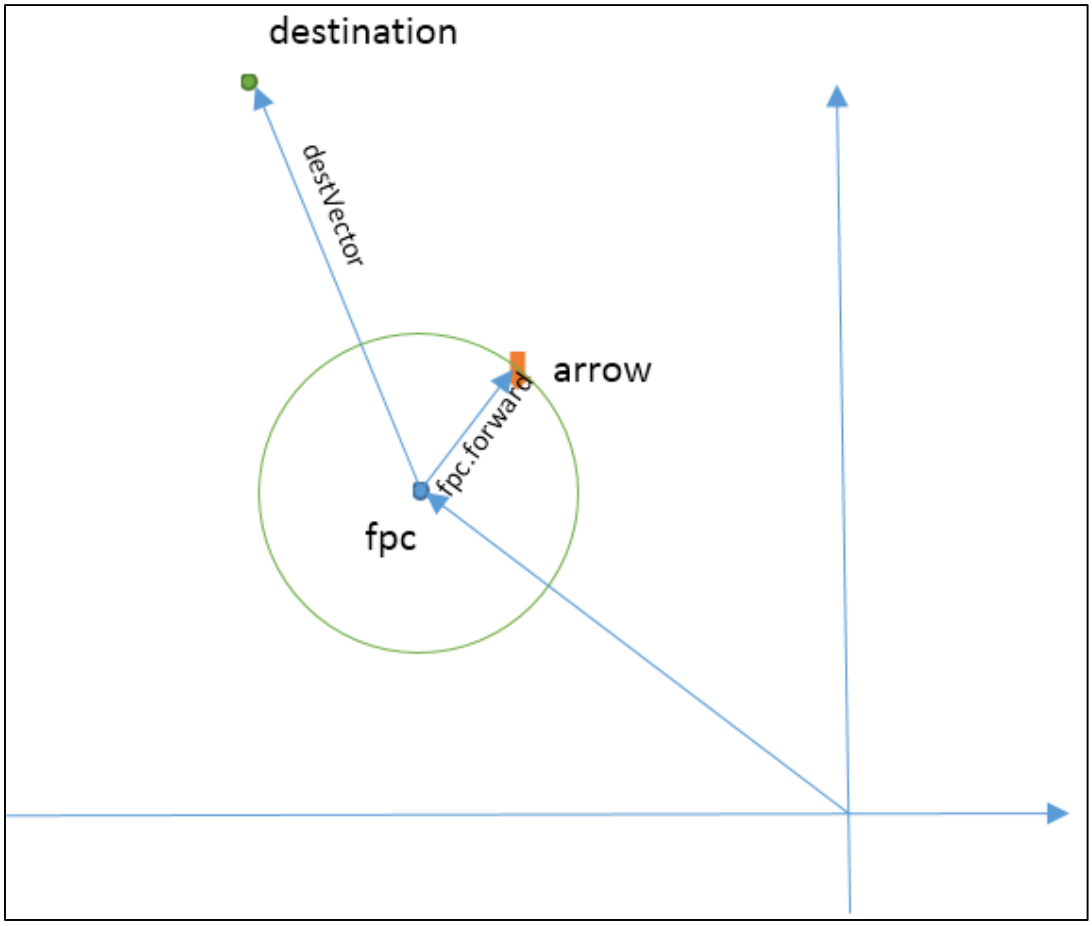

Рисунок 3.17 — Зображення вектора між користувачем та кінцевою точкою призначення

Далі потрібно розрахувати кут між знайденим вектором і віссю z, взявши за вісь обертання вісь y (3.5). Тут слід зазначити декілька моментів:

— має бути порахований кут зі знаком (+ або -); інакше стрілка правильно показуватиме тільки в деяких випадках;

— unity має спеціальну зручну функцію Vector3.SignedAngle, що і дозволяє порахувати такий кут. Саме тому і формула буде наведена, зважаючи на це. В іншому випадку кут між векторами можна було б порахувати, наприклад, використовуючи скалярний добуток векторів.

$$sAngle = -signedAngle(\overline{destVector}, \overline{zAxis} * cameraRotationDrift, \overline{yAxis}),(3.5)$$

де $sAngle$ — шуканий кут;

$\overline{destVector}$ — вектор, що сполучає користувача (spherePointer) та точку, куди потрібно дістатися (destination), порахований у попередній формулі;

$zAxis$ — вісь z;

$cameraRotationDrift$ — поворот системи координат після синхронізації з маркером;

$yAxis$ — вісь y.

Фінальний варіант малюнку — рисунок 3.18.

В формулі вище можна помітити нову змінну, а сама cameraRotationDrift. Після синхронізації з маркером глобальна система координат повертається, тому потрібно вносити деяку поправку, а саме поворот. Ця змінна і є таким поворотом. Як видно, вісь z домножається на неї. cameraRotationDrift представлений у вигляді кватерніону.

Далі, власне, використовуючи знайдену величину, і задається поворот. Це робиться за допомогою функції Euler класу Quaternion. Вона відповідає за переведення повороту, заданого в градусах, в тип кватерніону. Це необхідно, адже Unity оперує поворотами саме на базі кватерніонів. Кватерніон — чотирьохвимірне розширення множини комплексних чисел, тобто гіперкомплексне числа. Зараз

кватерніони широко використовуються у комп'ютерних технологіях для опису обертання об'єктів.

Слід лише додати, що по осі x поворот зроблений на -90 градусів, аби стрілка повернулася паралельно підлозі, адже від самого початку стрілка завантажується в сцену вертикально.

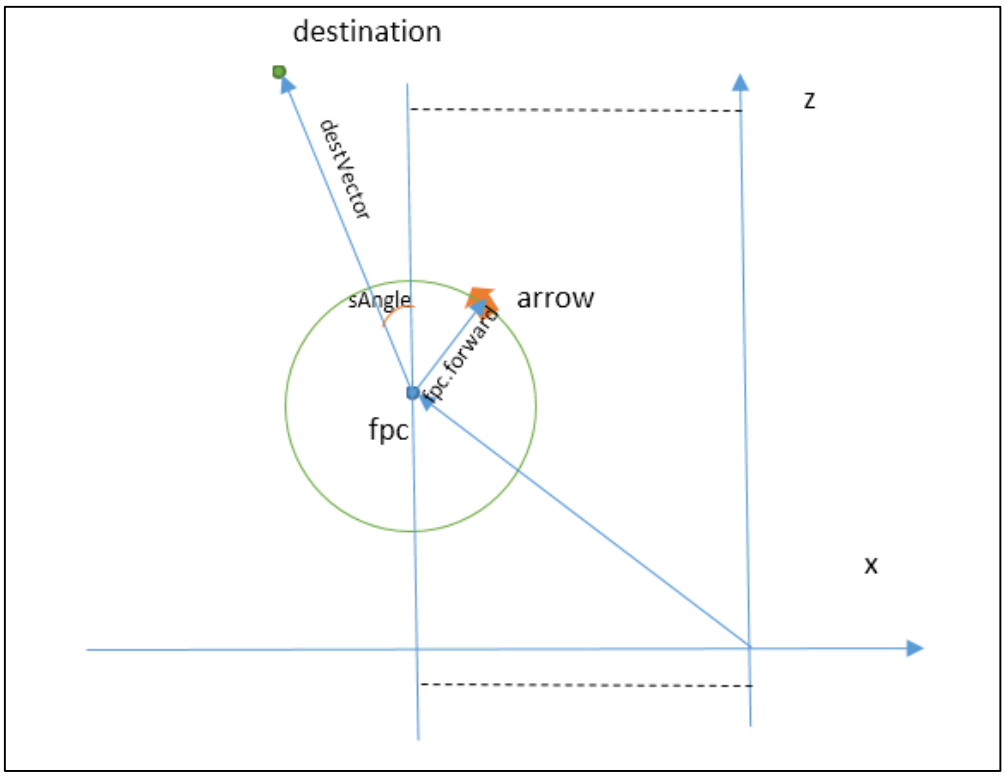

Рисунок 3.18 — Зображення кута, розрахунок якого необхідний для задання повороту стрілки.

Після таких розрахунків і дій стрілка завжди з'являтиметься перед користувачем, вказуючи своїми поворотом напрямок руху.

3.2 Опис та використання NavMesh

Системі навігації потрібні власні дані для представлення областей на сцені гри, по яких агент зможе ходити. Такі області визначають місця в сцені, де агент може стояти і рухатися. В Unity агенти описуються як циліндри. Область будується

автоматично з геометрії в сцені шляхом випробування місць, де може стояти агент. Потім місця з'єднуються з поверхнею, що лежить поверх геометрії сцени. Ця поверхня називається навігаційною сіткою (коротко NavMesh) [13].

NavMesh зберігає поверхню у вигляді опуклих багатокутників. Так як між двома точками всередині багатокутника перешкод немає, подання цих областей у вигляді опуклих багатокутників є вдалим і зручним. Також зберігається інформація про те, які багатокутники є сусідніми, що дозволяє розглядати всю зону в цілому для переміщень.

Для знаходження шляху на сцені між двома точками спочатку відбувається зіставлення точок початку та призначення з найближчими багатокутниками. Потім починається пошук від точки старту: алгоритм перевіряє всі сусідні багатокутники, поки не дістанеться до полігону призначення. Відвідані багатокутники відстежуються, що дозволяє знайти послідовність полігонів, яка веде від початку до пункту призначення. Загальний алгоритм пошуку шляху - це A* ("А-Стар"), який і використовується в Unity.

Коридор — це послідовність багатокутників, які описують шлях від початку до кінцевого полігону. Агент завжди йде в сторону наступного видимого кута коридору. Якщо програма проста, в якій на сцені рухається один агент, то гарна практика — це знайти всі кути коридору за один раз та анімувати персонажа так, щоб він рухався від кута до кута. Якщо агентів декілька, то задача трохи ускладнюється [14].

Положення наступного кута визначається логікою керування і, виходячи з цього, визначається бажаний напрямок і швидкість, необхідні для досягнення пункту призначення.

Агент балансує між запобіганням майбутніх зіткнень з краями навігаційної сітки (і іншими агентами) і рухом у потрібному напрямку. Це все тому, що механізм уникнення перешкод правильно обирає нову швидкість. Unity прогнозує та запобігає зіткненням.

Однією з найважливіших речей, яку слід зрозуміти щодо навігації, є різниця між глобальною та локальною навігацією (рисунок 3.19).

Глобальна навігація використовується для пошуку коридору на сцені. Пошук шляху на сцені вимагає досить багато процесорної потужності та пам'яті, тому це дорога операція.

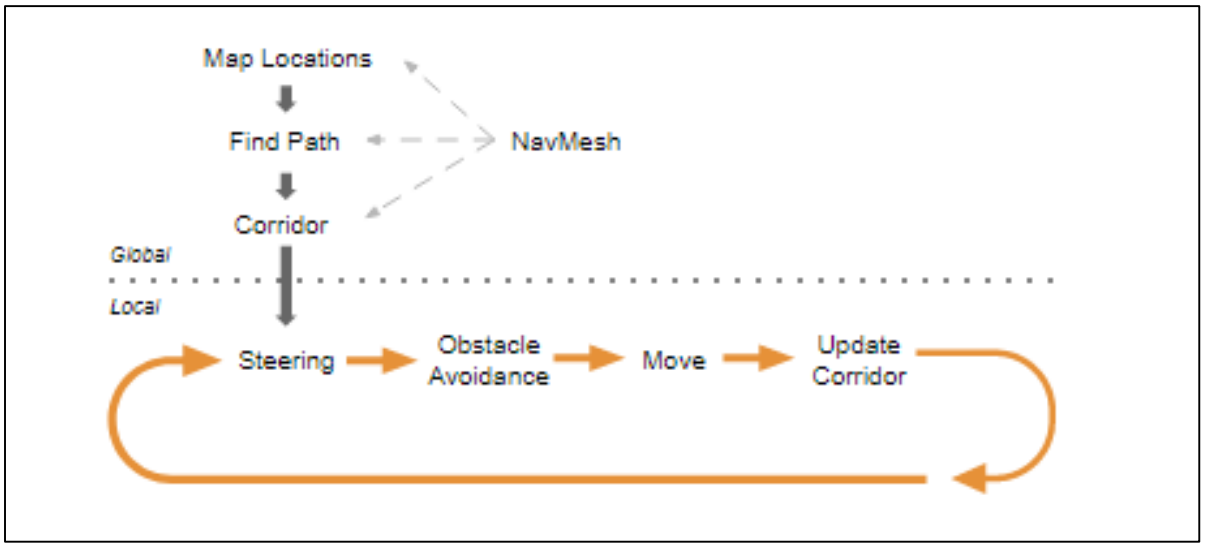

Рисунок 3.19 — Взаємодія локальної та глобальної навігації

Лінійний список багатокутників, що описують шлях, — це гнучка структура даних. Її можна локально регулювати по мірі того, як агент змінює позицію. Локальна навігація намагається зрозуміти, як, не стикаючись з перешкодами, ефективно рухатися до наступного кута.

Рухомі перешкоди найкраще подолати за допомогою локального уникнення перешкод. Таким чином агент може передбачувано уникнути перешкоди. На глобальну навігацію, тобто навігаційну сітку, може впливати перешкода, що стає нерухомою і вважається блокуванням шляху для всіх агентів.

Зв'язок між полігонами NavMesh описується з використанням посилань всередині системи пошуку шляхів. Іноді потрібно агентам дозволяти прохід в особливих місцях, де зазвичай проходу немає. Це, наприклад, перепригування перешкод. У такому випадку потрібно знати місце цієї дії.

Подібні дії помічаються за допомогою посилань Off-Mesh. Вони дають знати системі пошуку шляхів, що існує маршрут по вказаному посиланню. Потім

отримується доступ до цього посилання (під час проходження шляху, саме тоді ця дія може бути виконана.

Алгоритм A* (a-стар) — алгоритм пошуку, що знаходить найкоротший шлях. Також знайшов застосування в реальних ситуаціях, наприклад, на картах, в іграх, (там, де є багато перешкод) [15]. Якщо порівняти його з іншими алгоритмами, то алгоритм Дейкстри хороший в пошуку найкоротшого шляху, але він витрачає час на дослідження всіх напрямків, навіть безперспективних. Жадібний пошук досліджує перспективні напрямки, але може не знайти найкоротший шлях. Алгоритм A * використовує і справжню відстань від початку, і оцінку відстані до цілі [16].

Пропонується розглянути 2D сітку з перешкодами. Почнемо від початкової клітинки (червона внизу), щоб досягти цільової клітини (зелена внизу).

Розглянемо квадратну сітку з перешкодами, і на ній ми маємо вихідну клітинку та цільову клітинку (рисунок 3.20). На меті — досягнення цільової комірки (якщо можливо) зі стартової комірки за якомога найкоротший час. Тут на допомогу приходить алгоритм пошуку A*.

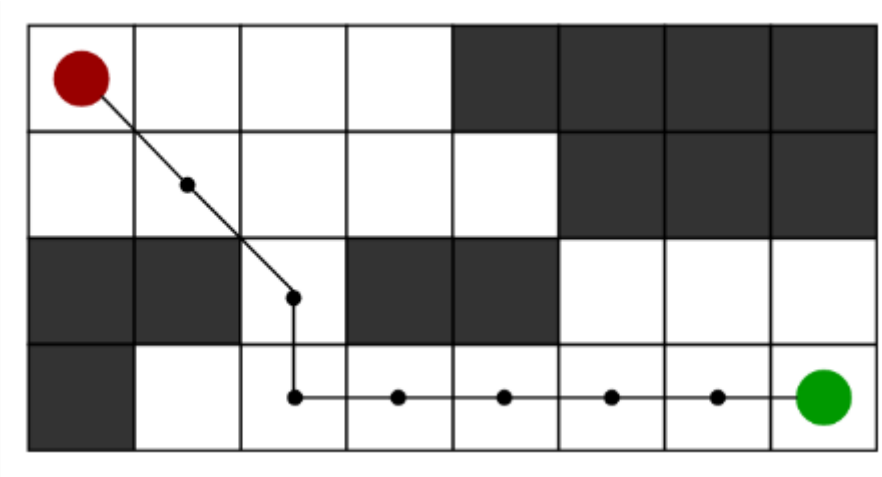

Рисунок 3.20 — Сітка з перешкодами, де треба знайти оптимальний шлях від початку до кінця

Алгоритм пошуку A* кожного кроку обирає комірку відповідно до значення параметра f, що дорівнює сумі двох інших параметрів (3.6). На кожному кроці він обирає комірку з найнижчим значенням f', і обробляє цей вузол [17].

$$f = g + h, \tag{3.6}$$

де f — евристична функція;

g — вартість переміщення до заданого квадрата в сітці від початкової точки;

h — функція евристичної оцінки відстані від початкової клітинки до кінця.

Функція h називається евристикою, що є чимось на кшталт розумної здогадки. Ми справді не знаємо фактичної відстані, поки не знайдемо шлях, тому що на шляху можуть бути всілякі перешкоди (наприклад, в іграх це можуть бути стіни чи вода тощо). Обчислити значення "h" можна багатьма способами.

Сам алгоритм можна описати наступним чином.

Ми створюємо два списки - відкритий та закритий.

а) Ініціалізувати відкритий список

б) Ініціалізувати закритий список

записати стартовий вузол у відкритий список (можна прийняти його f за нуль)

в). поки відкритий список не порожній

1) знайти вузол з найменшим f у відкритому списку, назвати його "q"

2) видалити q із відкритого списку

3) генерують 8 наступників q і встановлюють їх батька q

4) для кожного наступника

— якщо метою є наступник, припиніть пошук

successor.g = q.g + відстань між наступником і q

successor.h = відстань від цілі до наступника.

successor.f = successor.g + successor.h

— якщо вузол з тією ж позицією, що і наступник, знаходиться у відкритому списку, який має менше f, ніж наступник, пропустіть цього наступника

— якщо вузол з тією ж позицією, що і наступник, знаходиться у закритому списку, який має менше f, ніж наступник, пропустіть цього наступника, в іншому випадку додайте вузол до відкритого списку

кінець (для циклу)

5) додайте q до закритого списку

кінець (цикл)

Отже, припустимо, як на малюнку нижче, якщо ми маємо дістатися бажаної комірки з початкової комірки, то алгоритм пошуку A* піде шляхом, як показано нижче на рисунку 3.21. Слід зауважити, що наведений нижче малюнок зроблений з розглядом Евклідової відстані як евристики.

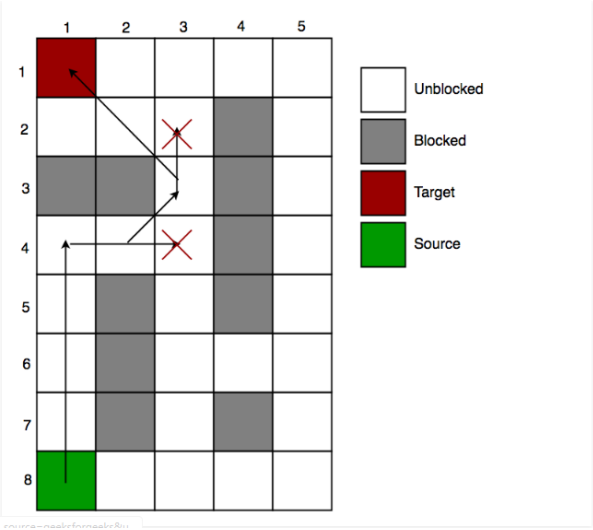

Рисунок 3.21 — Робота алгоритму A*

Ми можемо обчислити g. Далі пропонується розглянути варіанти обчислення h. Є такі варіанти:

— обчислити точне значення h (забирає багато часу);

— приблизна величина h з використанням деякої евристики (менш часозатратна);

Ми можемо знайти точні значення h, але це, як правило, дуже трудозатратно.

— попередньо перед запуском A* обрахувати відстань між кожною парою комірок;

— якщо немає заблокованих комірок / перешкод, існує варіант знаходження точного h без будь-яких попередніх обчислень, використовуючи формулу відстані / Евклідову відстань.

Для обчислення h зазвичай використовують три евристики наближення.

Першою з них є Мангеттенська метрика. Це сума абсолютних значень різниць координат x і y цільової клітинки та координат x та y поточної комірки відповідно (3.7).

$$h = abs(curretCell.x - goal.x) + abs(currentCell.y - goal.y) \qquad (3.7)$$

де h — функція евристичної оцінки відстані від початкової клітинки до кінця;

$curretCell$ — поточна клітина;

$goal$ — цільова клітинка

Цю евристику варто використовувати тоді, коли можна рухатися тільки у чотирьох напрямках.

Мангеттенська відстань показана на рисунку 3.22.

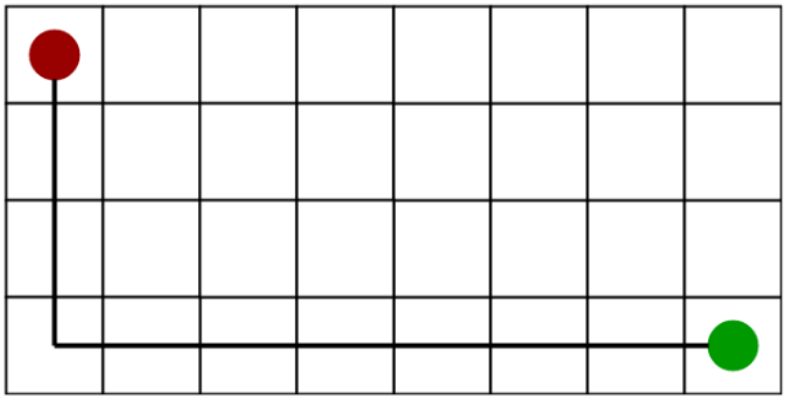

Рисунок 3.22 — Мангеттенська відстань

Наступною є діагональна відстань (рисунок 3.23). Це максимум абсолютних значень різниць координат x і y цільової та координатах x та y поточної комірки відповідно (3.8). Використовувати цей метод слід тоді, коли нам дозволяють рухатись лише у восьми напрямках

$$h = \max(abs(currentCell.x - goal.x, abs(currentCell.y - goal.y)) \quad ,(3.8)$$

де h — функція евристичної оцінки відстані від початкової клітинки до кінця;

$curretCell$ — поточна клітина;

$goal$ — цільова клітинка

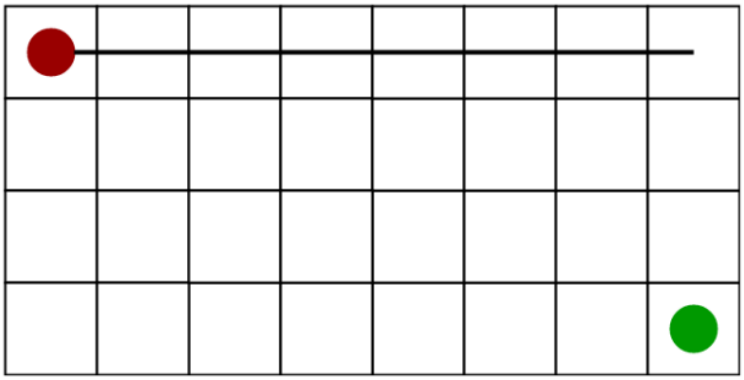

Рисунок 3.23 — Діагональна відстань

Останнім варіантом є евклідова відстань (риснок 3.24).

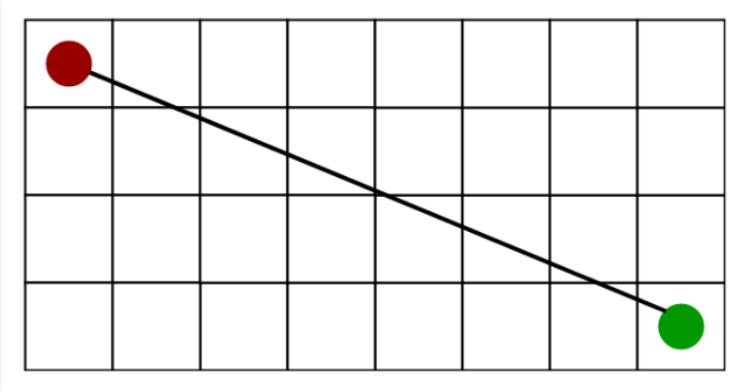

Рисунок 3.24 — Евклідова відстань

Як видно з назви, це відстань між поточною коміркою та цільовою коміркою, що вираховується за допомогою формули відстані (3.9)

$$h = sqrt\big((currentCell.x - goal.x) * 2 + (currentCell.y - goal.y)\big), \quad (3.9)$$

де h — функція евристичної оцінки відстані від початкової клітинки до кінця;

$curretCell$ — поточна клітина;

$goal$ — цільова клітинка

Використовувати слід тоді, коли нам дозволяють рухатися в будь-якому напрямку.

Якщо розглядати граф, є ймовірність, що потрібно буде пройти через всі ребра, щоб дістатися до кінцевої точки з вихідної. Отже, найгірша складність становить O (E), де E - кількість ребер у графі [18].

Висновки до розділу 3

В даному розділі були розглянуті тонкощі програмної реалізації підсистеми. Було показано місце програмного в модулю в системі. Наведено архітектуру системи, а також особливості розробки в середовищі Unity на прикладі даної підсистеми.

Крім того, було детально розглянуто організацію коду програмної підсистеми, описано підхід, за яким було реалізовано доповнену реальність, а також тонкощі використання Unity NavMesh.

4 ОПИС ВИКОРИСТАНИХ ПРОГРАМНИХ ЗАСОБІВ

Очевидно, що для розробки такого продукту, як мобільний додаток для навігації, потрібно мати встановленим на комп’ютері потужне середовище, що має зручний редактор, дозволяє легко оперувати об’єктами та задавати їх поведінку. Саме тому для даної роботи було обрано Unity — всесвітньо відомий і один з найпопулярніших ігрових рушіїв, що дозволяє також розробляти проекти для багатьох платформ.

Крім того, ще потрібен інструментарій, який дозволяє відслідковувати положення телефону в просторі і взаємодіяти з простором. ARCore від Google ідеально підходить для цієї задачі.

Звісно, потрібна і мова програмування для того, аби писати скрипти, які будуть визначати поведінку об’єктів додатку. Разом з Unity найчастіше використовується C#. Для написання та редагування коду, написаного цією мовою, було використано середовище розробки Microsoft Visual Studio 2017.

4.1 Ігровий рушій Unity

Ігрові рушії, такі як Unity, — це інструменти, що стоять за іграми. Unity — це одне із найпоширеніших та найпопулярніших середовищ для розробки ігор. Воно може використовуватися як і любителями, що розробляють свої перші найпростіші проекти, так і великими студіями, що займаються розробкою ігор і програм професійно. Юніті дозволяє створювати інтерактивні додатки та ігри для персональних комп’ютерів, ігрових консолей та мобільних телефонів [19].

Unity 3D — сучасний рушій. Його розробила компанія Unity Technologies.

Слід відмітити деякі важливі характеристики Unity. По-перше, для написання скриптів використовуються C# та JavaScript, що є набільш популярними мовами програмування. Також в Unity вбудований рушій гри, що надає можливість проводити тестування розроблюваної гри відразу в редакторі. Крім того, юніті

підтримує багато різноманітних форматів. З цією можливістю розробник може створювати моделі окремо в іншому середовищі, а розробляти продукт, в тому числі і програмувати, вже в Unity. Мінімальна умова для створення власної гри — це знання однієї з мов, що використовуються в Unity: C #, JavaScript або Boo.

Юніті — безкоштовний рушій. Обмеження — при запуску гри показується логотип Unity. З придбанням розширеної версії його можна вимкнути. Безкоштовність рушія — це те, що привернуло багатьох до розробки ігор з його використанням.

Unity — це рушій, який слід використовувати для проектів середнього розміру. Для маленьких ігор чи додатків використання такого серйозного рушія не виправдане. З іншого боку, в надто великих проектах краще застосувати інший рушій, адже Unity не надто швидкий через інтерпретовану мову C# та використання скриптів, що є його недоліком.

Ще одним недоліком є відносно великий розмір додатків.

Список платформ, що підтримує юніті, вражає. Програми, розроблені за допомогою Unity, можуть працювати практично всюди: на всіх операційних системах персональних комп'ютерів, на Андроїдта на iOS. Є також підтримка SmartTV, браузера. Також працює на незвичайних системах, таких як Tizen OS [20].

Потужний плюс Юніті — це ассети. Код, картинки — все представляється ассетами (Asset). Ассет може бути експортованим та імпортованим. Це надає змогу розробляти цілі ігрові шаблони, або заготовки. Залишиться тільки замінити картинки і трохи змінити скрипти.

Ще один плюс юніті — простота розробки додатків для мобільних телефонів. За допомогою рушія Unity розробляється величезна кількість мобільних ігор [21].

4.2 Платформа ARCore

ARCore — платформа, розроблена Google. Вона дозволяє створювати програми з використанням доповненої реальності.

ARCore надає можливість телефону взаємодіяти з інформацією, отриманою з навколишнього світу. Є 3 головні можливості ARCore для поєднання віртуальних елементів і реального світу (використовуючи камеру телефону):

— відслідковування руху дозволяє телефону зрозуміти та відстежувати його положення відносно реального світу;

— розуміння оточуючого середовища дозволяє телефону визначати розміри та розміщення різних поверхонь, як вертикальних, так і горизонтальних, таких як стіл або стіни;

— оцінка освітленості дає змогу телефону аналізувати умови освітлення середовища [22].

Коли переміщається телефон, ARCore будує всій власний світ на основі того, що його оточує. У ньому він розміщує віртуальні об'єкти. Також ним використовується технологія відстеження руху, що дає змогу визначити характер руху деяких об'єктів, зважаючи на рух камери [23].

Коли розробник в цьому просторі розміщує об'єкт, ARCore обраховує його положення відносно інших об'єктів. Після повернення в те саме місце об'єкт знову відображається.

На даний момент ARCore SDK доступний для:

— iOS;
— Unreal;
— Unity для Android;
— Unity для iOS;
— Android;
— Android NDK.

4.3 Інтегроване середовище розробки Microsoft Visual Studio

MicrosoftVisual Studio — це інтегроване середовище розробки, що надає змогу програмісту зручно створювати і редагувати програмний код. Інтегроване

середовище розробки (IDE) — це багатофункціональна програма, яку можна використовувати для різних моментів розробки програмного забезпечення. Разом з редактором і відладчиком, що доступні у багатьох інших середовищах розробки, Visual Studio також включає в себе компілятори, засоби виконання коду, графічні конструктори і багато інших функцій для спрощення процесу розробки програмного забезпечення [24].

Visual Studio підтримує всі найпопулярніші мови програмування: C, C++, VB.NET, C#, F#, JavaScript, Python.

Visual Studio підтримує багато зручних функцій для користувача, в тому числі підсвічування синтаксису. Істотно прискорити роботу допомагає технологія IntelliSense. Вона дозволяє автоматично завершувати код під час його написання [25].

Visual Studio має вбудований відладчик. Він може використовуватися для того, аби знайти і виправити помилки. Це можна робити навіть на низькому апаратному рівні. Оцінка якості коду стосовно використання пам'яті і продуктивності здійснюється за допомогою інструментів діагностики [26].

Найголовніші плюси Visual Studio:

— підтримує багато популярних мов;

— інтуїтивний стиль кодування (студія сама виділяє фрагменти коду кольором і розставляє необхідні відступи і т.п.);

— швидкість розробки доволі висока (IntelliSense, інструменти, такі як пошук з заміною і т.п.);

— відладка.

4.4 Мова програмування C#

C# — об'єктно-орієнтована мова програмування. Вона вважається високорівневою. C# підтримує різні парадигми програмування — від функціонального до компонентно-орієнтованого програмування [27]. Ця мова створена інженерами, яких очолив Андерс Хейлсберг, в 1998-2001 роках в компанії

Microsoft як основна мова для Microsoft .NET. Компілятор з C# входить до стандартного інсталювання самої .NET, тому програми на C# можна розроблювати і компілювати і без різноманітних середовищ, таких як Visual Studio [28].

Цій мові властиві гнучкість, потужість і універсальність. Програмісти пишуть з її використанням все — від невеличких додатків до серйозних програмних систем. Велика кількість фреймворків і бібліотек, зручний і звичний для багатьох сі-подібний синтаксис роблять цю мову такою універсальною і популярною [29].

Платформа .NET поставлялася завжди не безкоштовно. Це перешкоджало використанню мови, в тому числі і в професійних колах, що веде до низької її популярності. Але в листопаді 2014 Майкрософт почала забезпечувати безкоштовними ліцензіями з відкритим вихідним кодом всіх бажаючих.

Висновки до розділу 4

У розділі було наведено та описано основні засоби розробки програмного продукту.

Для розробки та налаштування проекту було використано ігровий рушій Unity. Він забезпечує зручність та швидкість розробки. Крім того, він сумісний також з іншим компонентом, а саме AR фреймворком ARCore. Він дозволив увести в розроблювану підсистему доповнену реальність, що покликана спростити процес навігації всередині будівлі, зробити його більш цікавим та інтерактивним.

Код скриптів було написано за допомогою об’єктно-орієнтованої мови програмування C# з використанням інтегрованого середовища розробки Microsoft Visual Studio.

Такий набір інструментів надає розробнику гарні можливості для написання серйозних проектів.

5 МЕТОДИКА РОБОТИ КОРИСТУВАЧА З ПРОГРАМНОЮ СИСТЕМОЮ

Інтерфейс розробленої програми є дружнім до користувача і інтуїтивно простим. Але для максимально продуктивної роботи рекомендовано ознайомитися з інсталяцією та сценарієм роботи програми.

5.1 Системні вимоги та інсталяція

По-перше, для використання розробленого програмного продукту потрібно мати android-пристрій, шо підтримує ARCore. Переглянути список таких пристроїв можна за посиланням: https://developers.google.com/ar/discover/supported-devices

По-друге, якщо користувач має при собі необхідний пристрій, йому необхідно завантажити на телефон файл з розширенням *.apk, після чого, використовуючи його, встановити систему навігації.

5.2 Сценарій роботи користувача з підсистемою

Після синхронізації з маркером і визначення місцезнаходження користувача (це інший модуль системи) користувач вже може здійснювати навігацію корпусом. Для початку навігації необхідно обрати кінцеву точку призначення, тобто місце або аудиторію, до якої йому потрібно дістатися.

Обрати пункт призначення можна двома способами.

— Якщо пункт призначення знаходиться на тому поверсі, що і користувач, то доступна можливість обрати будь-яке місце на цьому поверсі. Для цього потрібно доторкнутися до бажаної точки призначення і затримати дотик на секунду. Після цього з’явиться оптимальний шлях, який вестиме користувача до пункту призначення (рисунок. 5.1).

— Існує можливість обрання конкретної аудиторії як точки призначення. Для цього потрібно доторкнутися до випадаючого списку, що в верхньому правому кутку екрана. Обрати аудиторію можна з будь-якого доступного поверху (рисунок 5.2).

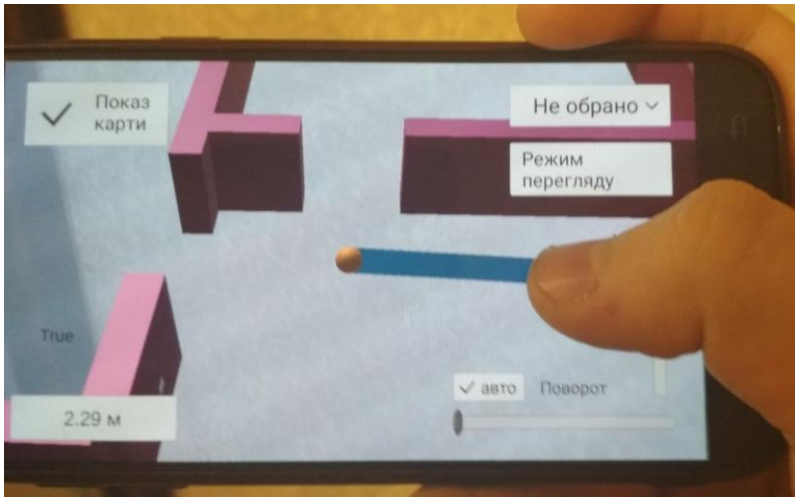

Рисунок 5.1 — Вибір точки призначення за допомогою довгого натискання

Після цього користувачу на вибір доступні 2 режими, в яких може здійснюватися навігація.

За замовчуванням система знаходиться в першому режимі, в якому відображується поточний поверх, шлях до точки призначення, тобто карта, що видно з рисунку вище.

Користувач може перемкнутися в інший режим. Це робиться за допомогою кліку на перемикач "показ карти". Якщо зняти галку, то шлях разом з картою зникнуть. Натомість додаток буде перемкнутий в режим камери, і на екрані з’явиться червона стрілка, яка буде розташована начебто прямо перед камерою. В яку б сторону користувач не направляв камеру, стрілка завжди буде розташована перед нею. При цьому вона буде вказувати шлях до найближчої контрольної точки всієї траєкторії. (рисунок 5.3)

В будь-якому з цих режимів у нижньому лівому кутку екрана відображається відстань до найближчої контрольної точки на шляху та до кінцевої точки призначення. Це можна побачити на рисунках вище.

Якщо користувач знаходить на одному поверсі, а кінцева точка призначення — на іншому, то система приведе його до найближчих сходів, після чого на екрані

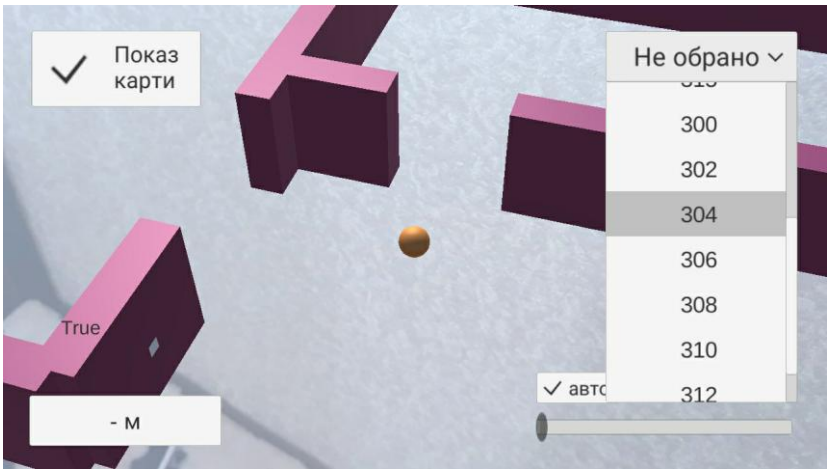

Рисунок 5.2 — Вибір точки призначення за допомогою випадаючого списку

з’явиться повідомлення, яке проінструктує, що робити далі: спускатися чи підніматися та на скільки поверхів. (рисунок 5.4)

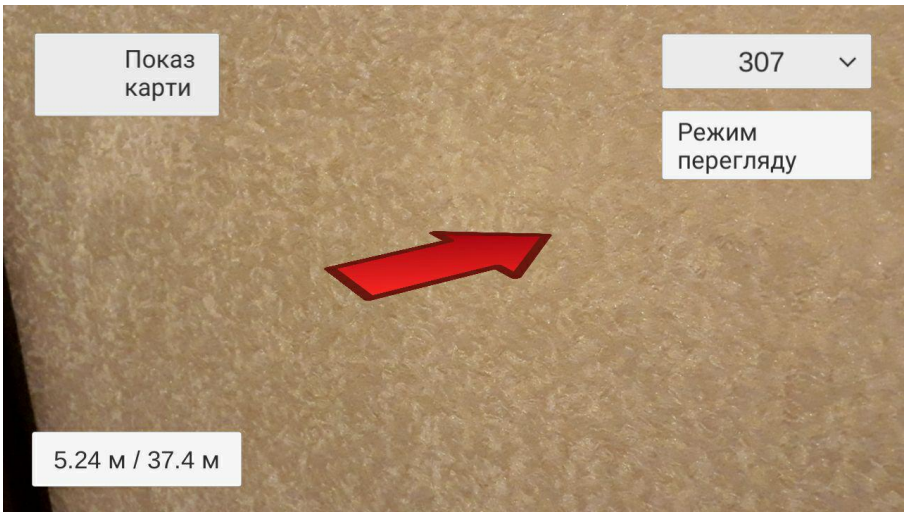

Рисунок 5.3 — Стрілка в режимі камери, що вказує правильний шлях

Якщо система знаходиться в режимі карти, то доступний режим перегляду всього поверху. За замовчуванням він вимкнений. Щоб його увімкнути, треба поставити галку в перемикачі "режим перегляду", що в правій частині екрану.

В цьому режимі також можна ставити кінцеву точку призначення. Крім того, внизу екрану є 2 слайдери. Горизонтальний відповідає за поворот карти. Якщо стоїть галка "авто", то це означає, що карта сама повертається по мірі того, як користувач рухає свій телефон.

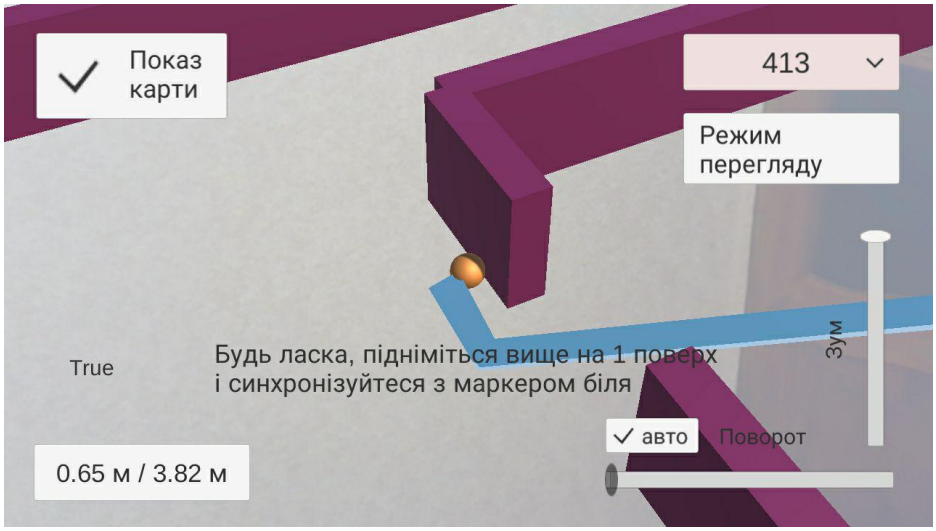

Рисунок 5.4 — Повідомлення про перехід на інший поверх

Якщо ж цю галку зняти, то слайдером можна повертати карту так, як зручно користувачу. (рисунок 5.6).

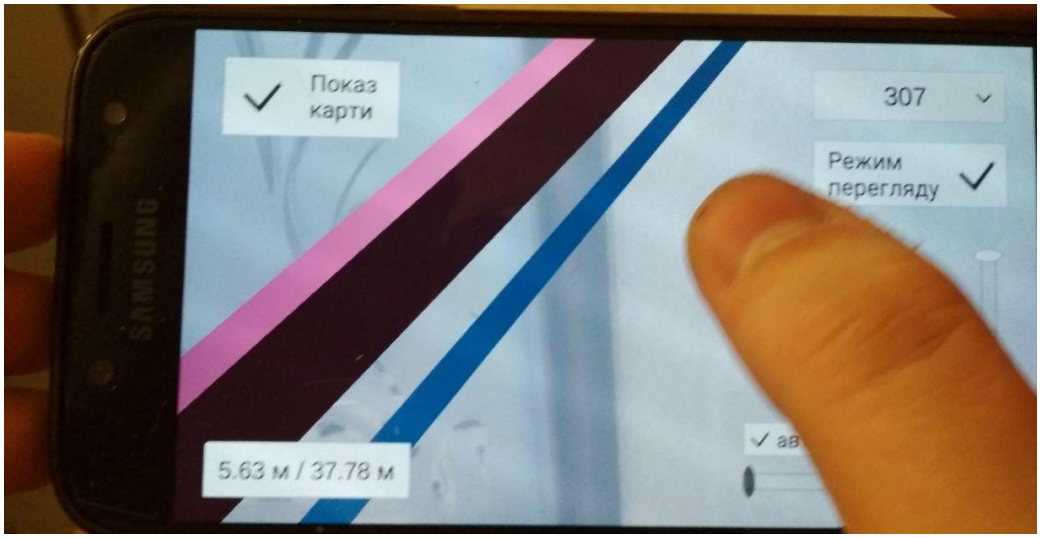

Рисунок 5.5 — Перегляд частини карти, якої у звичайному режимі не видно

Якщо після цього поставити назад галку "авто", то камера повернеться у вихідне положення.

Інший слайдер відповідає за зум (рисунок 5.7).

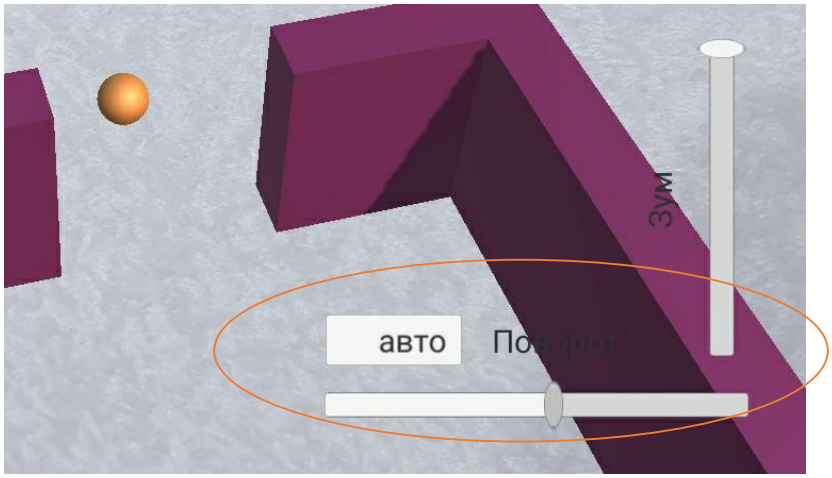

Рисунок 5.6 — Користувач сам повертає карту за допомогою відповідного слайдеру

Тобто з його використанням можна або детальніше, ближче розглянути якусь частину карти, або навпаки відобразити більше карти на екрані.

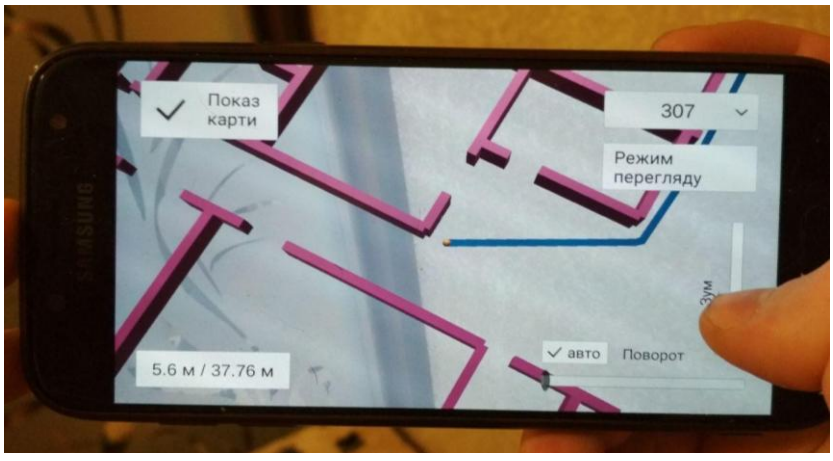

Рисунок 5.7 — Користувач використовує слайдер, щоб відобразити більше карти на екрані (слайдер зуму)

Також зум можливо здійснити іншим способом. Для цього потрібно поставити два пальця на екран, а потім або зсовувати їх докупи, щоб віддалити зображення, або розсовувати їх один від одного, щоб наблизити карту (рисунок 5.8). Такий спосіб є загальноприйнятим і використовується в багатьох інших додатках, тому його було реалізовано і в даній програмі.

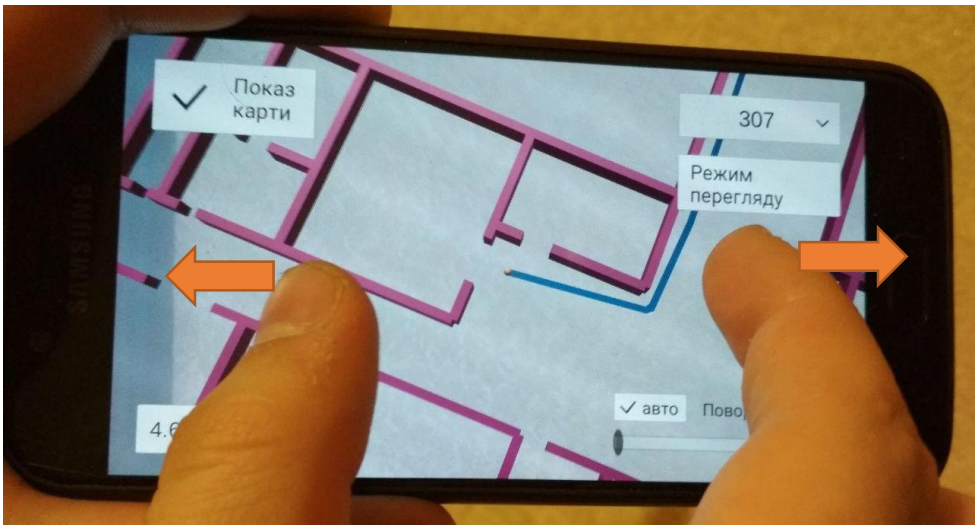

Рисунок 5.8 — Використання опції зуму за допомогою розсовування пальців

Слід зауважити, що в режимі камери/стрілки режим перегляду недоступний, адже карта в цей час відключена.

Висновки до розділу 5

В даному розділі були описані системні вимоги до програмного продукту, а також процедура його інсталяції, що передбачає завантаження на телефон файлу та його встановлення.

Крім того, було наведено детальну інструкцію по роботі з системою для користувача. Це означає, що, ознайомившись з нею, він буде максимально готовий до робити з програмою, а ймовірність появи питань стосовно її використання буде мінімальною.

6 РОЗРОБКА СТАРТАП ПРОЕКТУ

З самого початку ця розроблювана система задумувалася як стартап-проект. Він полягає в розробці продукту, що має забезбечувати навігацію в будівлі, використовуючи доповнену реальність, і який при цьому має доступну ціну. Крім того, він не має використовувати ніяких додаткових пристроїв, крім телефону, не працювати з базою даних, що б додатково навантажило телефон. Таким чином, система має розгортатися навіть на не дуже потужних пристроях.

Звісно, для початку потрібна ідея; коли вона готова, то для того, аби стартап-проект був успішний, треба оцінити всі ризики і загрози, порівняти майбутній продукт з розробками конкурентів, оцінити його слабкі і сильні сторони, розглянути потенційних покупців та оцінити, наскільки ринок готовий до такого продукту. Саме це і буде описано в цьому розділі.

6.1 Опис ідеї проекту

Будь-який проект починається з ідеї. Розробка даної системи — не вийняток. Саме тому у таблиці 6.1 описана ідея розробки стартап-проекту.

Таблиця 6.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Додаток навігації всередині приміщення для смартфонів з використанням доповненої реальності	1. Навігація в навчальних корпусах університетів, шкіл і т.п.	Швидке знаходження необхідної аудиторії; особливо актуально для нових студентів або учнів.

Таблиця 6.1 (продовження)

	2. Невеликі торгові центри та магазини	Дозволяє зекономити час і дістатися до необхідного відділу магазину чи товару швидше; для власника магазину (непрямий користувач) це означатиме збільшення потоку користувачів, а також введення реклами товарів за бажанням.

Як видно з таблиці вище, для розробленого продукту напрямками застосування обрано використання в таких установах, як торгові центри невеликого розміру та навчальні заклади (будівлі відділень, навчальні корпуси і т.п.). Очевидними в такому випадку є вигоди для користувачів: у першому випадку студенти будуть відразу знаходити аудиторії і встигатимуть на заняття вчасно, і викладачі їх не чекатимусть; у другому випадку кінцеві користувачі (покупці) зможуть швидко і зручно діставатися до необхідних їм відділів, а власник магазину придбанням такого додатку збільшить при цьому потік користувачів, а також зможе ввести рекламу деяких товарів у додаток.

Наступним кроком є визначення характеристик продукту, що планується розробити (таблиця 6.2).

З таблиці 6.2 зрозуміло, що відбувається порівняння з деякими конкурентами. За продукти конкурентів було обрано ті, що і в розділі 2 в пункті 2.2. Конкурент 1 — Inplaces, 2 — Hubbell IPS, 3 — IndoorAtlas.

Таблиця 6.2. Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№ п/п	Техніко - економічні характеристики ідеї	(потенційні) товари/концепції конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
		Розроблюваний проект	Конкурент 1	Конкурент 2	Конкурент 3			
1	Точність	середня	висока	висока	низька		+	
2	Собівартість	низька	середня	висока	висока			+
3	Використання додаткового обладнання	Ні	Ні	Так	Так			+
4	Кросплатформенність	Ні	Так	Так	Так	+		
5	Використання доповненої реальності	Так	Так	Ні	Ні			+

Аналізуючи цю таблицю, стає зрозуміло, що сильними сторонами розроблюваного продукту є відсутність використання додаткового обладнання і наявність доповненої реальності, а також низька ціна. На жаль, продукт не є кросплатформним, що є його слабкістю. Слід додати, що програма має непогану точність, якої вистачить для невеликих і середніх приміщень.

На основі цих міркувань можна стверджувати конкурентноспроможність продукту.

6.2 Технологічний аудит ідеї проекту

В цьому пункті слід провести аудит технології проекту, що має допомогти реалізувати ідею (технології створення товару).

Перший крок для цього — це визначення технологій, які можуть використовуватися в процесі розробки (таблиця 6.3).

Таблиця 6.3. Технологічна здійсненність ідеї проекту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Елементи доповненої реальності	ARCore	Наявна	Доступна
		ARKit	Наявна	Доступна
		Wikitude	Наявна	Платна
2	Пошук шляху (Pathfinding) і навігація картою	Unity NavMesh	Наявна	Доступна
Обрана технологія реалізації ідеї проекту: ARCore, NavMesh				

З наведеної таблиці видно, що для додання в проект елементів доповненої реальності можуть бути використані кілька фреймворків. ARCore довзоляє розгортати додатки на пристроях, що працюють на базі Android, але також є ймовірність, що програма запуститься і на iOS. ARKit — це технологія Apple, розроблена спеціально під iOS. Wikitude — ще AR фреймворк, проте він платний, і його обрання збільшить собівартість продукту. При цьому якість та можливості ARCore не поступаються Wikitude. До того ж, він безкоштовний. Саме тому з трьох аналогів було обрано ARCore.

Для реалізації пошуку шляху і навігації картою було обрано найпопулярнішу для цього технологію — Unity NavMesh.

6.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей, які можна використати під час ринкового впровадження проекту, та ринкових загроз, які можуть перешкодити реалізації проекту, дозволяє спланувати напрями розвитку проекту із урахуванням стану ринкового середовища, потреб потенційних клієнтів та пропозицій проектів-конкурентів [30].

Спочатку варто провести аналіз попиту (таблиця 6.4).

Таблиця 6.4. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристка
1	Кількість головних гравців, од	Біля 5
2	Загальний обсяг продаж, грн/ум.од	25 000 грн за ум.од.
3	Динаміка ринку (якісна оцінка)	Очікується зростання
4	Наявність обмежень для входу (вказати характер обмежень)	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності в галузі (або по ринку), %	15

Аналізуючи попередню таблицю, зрозуміло, що вкладання коштів в цей проект є вигідним. Слід звернути увагу, що обмежень для входу і до стандартизації немає, що полегшує ситуацію. До того ж, на ринку очікується зростання в цій сфері.

Наступним кроком є визначення цільової аудиторії програмного продукту, її поведінки та вимог (таблиця 6.5).

Таблиця 6.5. Характеристика потенційних клієнтів стартап-проекту

№ п/п	Потреба, що формує ринок	Цільова аудиторія	Відмінність у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Можливість швидко і зручно здійснювати навігацію будівлею	Учні, студенти	Не передбачається	Зручність інтерфейсу, швидкість роботи, можливість швидко дістатися до аудиторії
2		Покупці в торгових центрах та схожих структурах	Не передбачається	Зручність інтерфейсу, швидкість роботи, можливість швидко дістатися до місця призначення, достатньо велика точність (через те, що будівлі подібного характеру зазвичай більші)

Як видно з попередньої таблиці, цільовою аудиторією є студенти в навчальних закладах та покупці в торгових центрах. При цьому відмінності в поведінці цих двох цільових аудиторій не очікується: всі вони бажають одного і того ж — швидкої і якісної навігації. Щоправда, вимоги у відвідувачів торгівельних центрів можуть бути трошки вищі, адже через більшу площу будівлі і точність роботи додатку має бути більша, щоб помилка, що нагромаджується, була якнайменшою.

Наступним кроком є аналіз ринкового середовища. Необхідно в табличному вигляді визначити фактори загроз (таблиця 6.6) і можливостей (таблиця 6.7), що перешкоджають і допомагають впровадженню програми.

Таблиця 6.6. Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Ризик технології	Навігація без використання додаткового обладнання згодом може стати недостатньо точною для певного прошарку клієнтів	Підвищення точності за рахунок введення якого одного типу додаткових засобів (наприклад, Wi-Fi)
2	Недоступність платформи	Неможливість використання програмного забезпечення на пристроях, що не запускаються на базі андроїд	Розробка програмного забезпечення для інших платформ (наприклад, iOS).

В таблиці було наведено основні загрози виходу на ринок розроблюваного продукту. На думку розробників, найголовнішою з них є ризик технології. Так, раніше згадувалося, що відмова від використання додаткового обладнання, з однієї сторони, сприяє низькій собівартості продукту і легкості його розгортання. З іншої сторони, для деяких користувачів це може означати недостатню точність навігації, що зменшить коло потенційних користувачів. Якщо таке траплятиметься часто,

команді розробників варто, наприклад, удосконалити свій продукт, використавши якийсь один тип додаткових засобів.

Також існує загроза, що програний продукт не буде запускатися на платформах, що працюють не на базі android. В такому випадку є можливим варіант розроблення такого ж програмного забезпечення, але вже підлаштованого під інші платформи.

Таблиця 6.7. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Будування шкіл, вищих учбових закладів, торгових центрів або розширення існуючих	Існує можливість, що при збільшенні потенційних покупців (структур, яким може знадобитися продукт), збільшить і попит на забезпечення	Посилення рекламної кампанії

Найголовнішим фактором можливостей є будування нових торгових центрів, що може збільшити кількість потенційних користувачів.

Наступним кроком є визначення типу конкуренції (Таблиця 6.8)

Таблиця 6.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика
Тип конкуренції	Олігополія
За рівнем конкурентної боротьби	Глобальна

Таблиця 6.8 (продовження)

За галузевою ознакою	Міжгалузева
Конкуренція за видами товарів	Товарно-видова
За характером конкурентних переваг	Нецінова
За інтенсивністю	Не марочна

Такий аналіз конкуренції є непоганим, але неповним. Саме тому варто провести аналіз конкуренції в галузі за М. Портером (таблиця 6.9).

Таблиця 6.9. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачаль-ники	Клієнти	Товари-замінники
	Навести перелік прямих конкурентів	Визначити бар'єри входження в ринок	Визначити фактори сили постачаль-ників	Визначити фактори сили споживачів	Фактори загроз з боку замінників
Висновки:	Існує близько 5 найвпливовіших конкурентів. Конкурент 1 найсерйозніший:використовує схожий підхід і технології	Можливості виходу на ринок є, як і у конкурентів. Строки виходу їх продукту на ринок — приблизно 1 рік	Постачальники відсутні	Зручність, достатня точність	Загроза розробки конкурентами більш точного та (або) дешевого аналогу

Попередня таблиця нам дає зрозуміти, що продукт має одного найсерйознішого конкурента, що має аналогічний підхід рішення ситуації. Врахувавши зручність і точність розроблюваного програмного продукту, а також відмову від використання додаткових пристроїв, можна вважати, що він має можливість роботи на ринку.

Використовуючи ці висновки, можна виокремити основні фактори конкурентоспроможності продукту (Таблиця 6.10).

Таблиця 6.10. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентноспроможності	Обгрунтування
1	Вартість	Передбачається випуск товару з ціною, нижчою ніж у конкурентів
2	Функціонал	Наявність режиму з доповненою реальністю, що додає цікавості і зручності у використанні, а також режиму перегляду карти
3	Не потребує додаткового обладнання	Для роботи програми не потрібні, наприклад, Bluetooth beacons

З таблиці ми бачимо, що відмова від дод. обладнання, режим з доповненою реальністю і невисока вартість продукту (у порівнянні з конкурентами) є основними факторами, що дають перевагу над конкурентами.

Після таких висновків вже можна проаналізувати сильні та слабкі сторони проекту у порівнянні з конкурентами (таблиця 6.11).

Посилаючись на цю таблицю, ми бачимо, що найбільше балів набрано за фактором "вартість". В двох інших категоріях розроблюваний програмний продукт також має перевагу.

Заключною таблицею ринкового аналізу можливостей впровадження розробленого програмного продукту є аналіз SWOT.

Таблиця 6.11. Порівняльний аналіз сильних та слабких сторін проекту

№ п/п	Фактор конкурентноспроможності	Бали 1-20	Рейтинг товарів-конкурентів у порівнянні з розроблюваним продуктом						
			-3	-2	-1	0	1	2	3
1	Функціонал	17			+				
2	Вартість	19	+						
3	Не потребує дод. обладнання	17			+				

SWOT представляє собою сильні та слабкі сторони, можливості і загрози для проекту в одній таблиці (таблиця 6.12).

Таблиця 6.12. SWOT-аналіз стартап-проекту

Сильні сторони: функціонал, вартість, не потребує дод. обладнання	Слабкі сторони: Недоступність на платформах, що не підтримують ARCore, недостатня точність для деяких клієнтів
Можливості: Продаж більшої кількості товару через збільшення навчальних закладів, торгових центрів Індивідуалізація програми під конкретного замовника Гібридизація технологій для підвищення точності	Загрози: віддання переваги потенціальними клієнтами продуктам конкурентів через недоступність інших платформ та технологічний ризик (не використовуються дод. апаратура)

Вищенаведена таблиця по суті є узагальненням вищесказаного. Тут варто додатково пояснити тільки кілька моментів.

Якщо клієнти віддаватимуть перевагу продуктам конкурентів через недостатню точність (техн. ризик), то, можливо, варто впровадити гібридизацію технологій, що означає залучення якогось одного типу дод. обладнання до роботи (наприклад, Wi-Fi). Індивідуалізація програми під конкретного замовника може бути виконана, наприклад, у вигляді додання до програми реклами (у випадку з торговим центром), зміни кольорової гами за бажанням клієнту і так далі.

Використовуючи SWOT-аналіз, можна скласти таблицю 6.13 альтернатив ринкового впровадження стартап-проекту.

Таблиця 6.13. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива	Ймовірність отримання ресурсів	Строки реалізації
1.	Розробка системи навігації з використання додаткового обладнання (Bluetooth Beacons)	20%	1,5
2.	Розробка системи навігації з використанням тільки комп’ютерного зору	90%	1

Враховуючи вищенаведену таблицю, зрозуміло, що для реалізації стартап-проекту доцільніше взяти другу альтернативу, а саме розробку системи навігації з використанням тільки комп’ютерного зору, адже, по-перше, на це піде менше часу, по-друге, не слід забувати, що стартап-проект є ще і дипломною роботою, тому при

обранні першої альтернативи очікуються проблеми з наданням коштів для додаткового обладнання, тому раціональніше обрати інший спосіб.

6.4 Розробка ринкової стратегії проекту

В розробленні ринкової стратегії перше, що треба зробити, це визначити стратегію охоплення ринку (таблиця 6.14).

Таблиця 6.14. Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Навчальні заклади	+	Середній	Практично немає	Просто
2	Торгові центри	+	Високий	Вище середнього	Вище середнього
3	Великі людні установи, наприклад, аеропорти	+	Вище середнього	Висока	Складно
Які цільові групи обрано: 1, 2					

Переглядаючи таблицю вище, зрозуміло, що за цільові групи користувачів обрано навчальні заклади та торгові центри. Хоч попит є у всіх трьох категорій, слід зауважити, що через відсутність конкуренції в навчальних закладах цю категорію найпростіше охопити. Також можна спробувати запропонувати даний продукт

торговим центрам, хоч там і досить велика конкуренція. Для останньої категорії даний продукт підходить не зовсім: для таких величезних за площею установ треба продукт, що має більшу точність.

На основі цих міркувань можна навести базову стратегію розвитку (таблиця 6.15).

Так як планується співпраця з кількома сегментами ринку, за стратегію охоплення ринку було обрано стратегію диференційованого маркетингу; до того ж,

Таблиця 6.15. Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Розробка системи навігації з використанням тільки комп’ютерного зору	Стратегія диференційованого маркетингу	Функціонал, Вартість, Не потребує додаткового обладнання	диференціація

деякі особливості продукту будуть змінюватись в залежності від сегменту і конкретного користувача. Базовою стратегією розвитку визначено диференціацію, адже товар пропонує не зовсім поширений підхід: доповнену реальність і відмову від додаткового обладнання.

Наступний крок — визначення базової стратегії конкурентної поведінки (таблиця 6.16)

Звісно, розроблений продукт не буде першопрохідцем на ринку: до нього вже існували системи навігації всередині будівлі; дуже рідко, але деякі навіть

використовували доповнену реальність. Найоптимальнішим варіантом буде пошук зовсім нових користувачів (наприклад, ті ж навчальні заклади). Неможливо створити зовсім нову систему навігації, відмінну від тих, що пропонують конкуренти, тому деякі характеристики будуть однакові.

Зважаючи на вимоги користувачів, стратегію розвитку та конкурентної поведінки, далі треба розробити стратегію позиціювання. Під цією стратегію розуміється формування ринкової позиції (комплексу асоціацій), що допоможе

Таблиця 6.16. Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект "першопрохідцем" на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки*
1	Ні	Шукати нових споживачів	Так	Стратегія наслідування лідеру

споживачам з ідентифікацією проекту (таблиця 6.17).

Ця таблиця нам яскраво демонструє комплексну позицію проекту, а саме: доступність, зручність, і функціональність, що сформована, виходячи з вимог користувачів з обраних сегментів і ключових конкурентоспроможних позицій проекту.

Таблиця 6.17. Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Дешевизна, зручність інтерфейсу, точність	Диференціація	Відмова від додаткового обладнання зробить продукт дешевшим, а режим з доповненою реальністю допоможе цікавіше і швидше знаходити бажане місце	Доступність, зручність, функціональність

6.5 Розроблення маркетингової програми

Розробляючи маркетингову програму, з самого початку потрібно сформувати маркетингову концепцію товару (таблиця 6.18).

Таблиця 6.18. Визначення ключових переваг концепції потенційного товару

п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Функціонал	Режим доповненої реальності	Наявність режиму, якого практично немає ні в одного з конкурентів
2	Вартість	Вигідна ціна для покупців	Нижча ціна, ніж у більшості конкурентів

Таблиця 6.18 (продовження)

3	Незалежність системи від дод. пристроїв	Не потребує додаткового обладнання	Потрібен тільки смартфон

Наступний крок — розробка трирівневої маркетингової моделі товару (таблиця 6.19).

Таблиця 6.19. Опис трьох рівнів моделі товару

<table>
<tr><td>Рівні товару</td><td colspan="3">Сутність та складові</td></tr>
<tr><td>I. Товар за задумом</td><td colspan="3">Додаток для смартфону, що допомагає здійснювати навігацію всередині будівлі, використовуючи доповнену реальність</td></tr>
<tr><td rowspan="4">II. Товар у реальному виконанні</td><td>Властивості/характеристики</td><td>М/Нм</td><td>Вр/Тх /Тл/Е/Ор</td></tr>
<tr><td>1. Низька ціна
2. Функціонал
3. Не потребує дод. обл.
4. Зручний інтерфейс</td><td>Нм</td><td>Тх</td></tr>
<tr><td colspan="3">Якість: продукт протестовано вручну</td></tr>
<tr><td colspan="3">Маркування: Немає</td></tr>
<tr><td></td><td colspan="3">Компанія: назва “Tecinnav”</td></tr>
<tr><td>III. Товар із підкріпленням</td><td colspan="3">Продаж товару за підпискою; постійна підтримка придбаного додатку</td></tr>
<tr><td colspan="4">За рахунок чого потенційний товар буде захищено від копіювання: обфускація коду</td></tr>
</table>

В попередній таблиці було наведено три рівні моделі товару. Це мобільний додаток, що пропонується для полегшення навігації всередині приміщення. Слід зауважити, що програма не купується тільки один раз, а доступна за підпискою. Це у свою чергу означає постійну і якісну підтримку придбаного програмного продукту. Товар буде захищено від копіювання та неправомірного розповсюдження за допомогою обфускації коду.

Далі потрібно визначити цінові межі (таблиця 6.20).

Таблиця 6.20. Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар
1.	30000	40000	350000	20000-25000

Після цього необхідно визначити систему збуту, оптимальну для даного проекту (таблиця 6.21).

Таблиця 6.21. Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1.	Покупка та щомісячна підписка	Продаж	0 (напряму)	Власна

Попередня таблиця презентує систему збуту. В її рамках прийняті такі рішення, що користувачі купуватимуть щомісячну підписку і що система збуту буде без посередників, власна, тобто глибина каналу збуту — 0.

Завершальним етапом формування маркетингової програми є розробка концепцій маркетингових комунікацій (таблиця 6.22).

Посилаючись на наступну таблицю, що показує концепцію маркетингових комунікацій для розробленого програмного продукту, зрозуміло, що в приміщення, де треба знайти необхідне приміщення і для якого розроблено даний додаток, буде

Таблиця 6.22. Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення
1.	Завантаження додатку через QR-коду, розміщеного в приміщенні.	Інтернет	Низька ціна, функціонал, не потрібно додаткового обладнання, зручний інтерфейс	Зацікавити потенційних користувачів; довести, що розроблений додаток економить час та допомагає спростити процес навігації.

розміщений QR-код. При його скануванні смартфон відкриє користувачеві сторінку, з якої можна буде завантажити додаток і здійснювати навігацію в рази простіше і зручніше.

Не зайвим також буде додати аналіз ефективності проекту. Далі він буде наведений на двох рисунках (рис. 6.1, рис. 6.2).

З рисунку нижче ми можемо побачити обраховані витрати на проект за рік. До постійних витрат слід віднести ті, які стабільно будуть витрачатися протягом року

Постійні	Змінні
Оренда приміщення - 20000	Витрати на збут – 2% - 500
Ком. послуги – 5000	Хостинг – 10% - 2600
Заробітня плата – 100000	
Маркетинг – 100000	Ремонт і обслуговування техніки – до 20000
Податки - 19500	
Всього: 244500	

Вартість ліцензії: $1000/міс + опціональні додаткові функції

СЗВ = $120=3200

ТБ $= 244500/(26000 - 120 * 26)$=11 ліцензій

Рисунок 6.1 — Витрати на проект

До них можна віднести оренду приміщення, комунальні послуги, заробітну плату розробникам, а також постійні витрати на маркетинг і податки.

• Ставка дисконтування: 40%/рік

Показник	0 рік	1 кв	2 кв	3 кв.	4 кв
1. Сума інвестицій, тис. грн.	-600	-	-	-	-
2. Виручка від реалізації, тис. грн.	-	260	875	1375	2250
3. Витрати на експлуатацію проекту, тис. грн.	-	766	840	900	1005
5. Ставка дисконту, %	-	10	10	10	10
6. Коеф. дисконтування	-	0.91	0.83	0.75	0.68
6. Грошові потоки, тис. грн.	-	-506	35	475	1245
7. Дисконтовані грошові потоки, тис. грн.	-	-460	29	356	847
8. NPV	-	-1060	-1031	-696	151

$$k_d = \frac{1}{(1+d)^t}$$

$$\text{ТО} = 3\text{кв} + \frac{696}{847}\text{кв}$$

3.82 кв=11.5 місяців

$$BCR = \frac{3524}{2752} = 1.28$$

$$P_c = \frac{BCR - 1}{t} = \frac{0.28}{4} = 7\%$$

Рисунок 6.2 — Аналіз ефективності

Змінні витрати — це витрати, які можуть змінюватися час від часу. До них слід віднести, наприклад, витрати на ремонт і обслуговування техніки.

Також було пораховано вартість однієї ліцензії, середні змінні витрати та точку беззбитковості, що дорівнює 11 ліцензіям.

Наступний рисунок показує, що було пораховано такі ключові показники, як виручка від реалізації, витрати на експлуатацію та ін. Також з нього зрозуміло, що аналіз ефективності було зроблено для чотирьох кварталів. При цьому ставку дисконтування було взято за 40%. Відповідно до розрахунків, проект має окупитися за 11.5 місяців.

Висновки до розділу 6

В рамках розділу були розглянуті головні моменти виходу на ринок запропонованого програмного продукту, що є мобільним додатком навігації всередині приміщень з використанням доповненої реальності.

Було запропоновано та детально розглянуто ідею для проекту, а також проведено технологічний аудит та аналіз ринкових можливостей запуску стартап-проекту. Після цього наступним кроком було розроблено ринкову стратегію проекту В кінці розділу було розроблено маркетингову програму для стартапу.

Для ринкової реалізації проекту було обрано альтернативу з використанням тільки комп’ютерного зору, при цьому обраним інструментом доповненої реальності є ARCore.

Крім того, було проаналізовано слабкі та сильні сторони проекту. До перших можна сміливо віднести відмову від використання додаткового обладнання, використання доповненої реальності та низьку цінову пропозицію, а до останніх — підтримка не на всіх смартфонах та недостатня точність для деяких користувачів. Було проаналізовано можливі ризики та запропоновано необхідні дії в разі їх прояву, в тому числі і гібридизація технології при технологічному ризику. Враховуючи це, було розроблено SWOT-аналіз.

В ході розробки програми було детально порівняно розроблюваний проект з аналогічними програмами конкурентів, такими як Inplaces, Hubbell IPS та IndoorAtlas. Порівняльний аналіз показав, що і в конкурентів, і в розроблюваній програмі є свої плюси та мінуси, проте в загальному остання має більш кращу і вигідну для

користувачів сукупність показників, маючи таку комплексну позицію: доступність, функціональність і зручність.

Було виявлено, що існує можливість комерціалізації такого проекту, адже показники ринку цьому сприяють (наприклад, є попит на подібні товари, очікується зростання ринку у цій сфері). Враховуючи потенційні групи клієнтів, невисокі бар’єри входження, доведену вище конкурентоспроможність продукту, можна зробити висновок про існування перспектив впровадження такого проекту.

ВИСНОВКИ

На початку роботи було проаналізовано існуючий процес навігації у більшості приміщень, в результаті чого було зроблено висновки про недосконаліть існуючих методів. Саме це послугувало причиною розробки всієї програмної системи загалом і модулю, створеного в ході виконання дипломної роботи, в тому числі, а також обгрунтувало доцільність і актуальність такої розробки.

Було проаналізовано проблеми створення даної підсистеми, головною з яких є вибір підходу використання доповненої реальності для розроблюваної програми, труднощі, пов’язані з різними варіантами, а також обрання інструментів, що допоможуть реалізувати обраний підхід.

Було створено підсистему проеціювання траєкторії руху на відображення об’єктів реального світу, що є частиною мобільного додатку для навігації корпусом. Він надає користувачу такі можливості:

— обрання бажаної точки призначення різними способами;
— навігація за допомогою карти;
— навігація з використанням режиму камери з доповненою реальністю;
— отримання інформації стосовно відстані, яку потрібно подолати;
— режим перегляду карти;
— додаткові опції.

Також було наведено тонкощі розробки підсистеми.

Програмний продукт було протестовано. Під час цієї перевірки він показав достатню точність для того, аби можна було здійснювати навігацію без проблем.

Крім того, в ході розробки програми було набуто досвіду роботи з Unity, а також було поглиблено знання стосовно мови програмування C#. Крім того було використано інструмент ARCore для додання в проект доповненої реальності, що додало знань як в цій сфері, так і у використанні цього пакету.

Спочатку розроблена система задумувалася до реалізації як стартап-проект. Саме тому в одному з розділів було охарактеризовано розроблений продукт саме з

цієї точки зору. Програмну систему було порівняно з розробками конкурентів, виділено її основні переваги. Також було розроблено і наведено документацію стосовно розробки такого проекту, починаючи від ринкової стратегії і закінчуючи маркетинговою програмою.

Отже, створений програмний продукт є важливим, адже дозволяє здійснювати переміщення корпусом більш організовано. Скоріше за все, подібна система була запропонована і реалізована вперше на кафедрі. Крім того, інтерфейс, як і загалом вся програма, є дружніми до користувача, що допоможе його швидкій адаптації до системи, а разом в інструкцією по роботі з програмою, запропонованою вище, дозволить максимально ефективно використовувати продукт.

Передбачається, що в майбутньому дана розробка допоможе більш зручно і швидко здійснювати навігацію всередині приміщень, особливо в корпусі ТЕФ.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Rehman U. Augmented Reality-based Indoor Navigation: A Comparative Analysis of Handheld Devices vs. Google Glass / U. Rehman, S. Cao. – 2017. Mealy
2. Mealy P. Virtual & Augmented Reality For Dummies / Paul Mealy. – Hoboken: For Dummies, 2018.
3. Jennifer C. A Review of Augmented Reality Applications for History Education and Heritage Visualisation / C. Jennifer, M. Minhua. – 2019.
4. Schmalstieg D. Augmented Reality: Principles and Practice / D. Schmalstieg, H. Tobias. – Boston: Addison-Wesley, 2016.
5. Azuma R. A Survey of Augmented Reality / Ronald Azuma. – 1997.
6. Projected Augmented Reality Intelligent Model of a City Area with Path Optimization / M.Mendes, J. Almeida, H. Mohamed, R. Giot. – 2019.
7. Ong S. Virtual and Augmented Reality Applications in Manufacturing / S. Ong, A. Nee. – London: Springer-Verlag, 2004.
8. Cawood S. Augmented Reality: A Practical Guide / S. Cawood, M. Fiala., 2008.
9. Linowes J. Augmented Reality for Developers: Build practical augmented reality applications with Unity, ARCore, ARKit, and Vuforia / J. Linowes, K. Babilinski., 2017.
10. Gerstweiler G. DARGS: Dynamic AR Guiding System for Indoor Environments / G. Gerstweiler, K. Platzer, H. Kaufmann. – 2017.
11. Glover J. Unity 2018 Augmented Reality Projects: Build four immersive and fun AR applications using ARkit, ARCore, and Vuforia / Jesseф Glover., 2018.
12. Jackson S. Unity 3D UI Essentials / Simon Jackson., 2015.
13. Buttfield-Addison P. Unity Game Development Cookbook: Essentials for Every Game / P. Buttfield-Addison, J. Manning, T. Nugent., 2019.
14. Baron D. Hands-On Game Development Patterns with Unity 2019: Create engaging games by using industry-standard design patterns with C# / David Baron. – Birmingham: Packt Publishing, 2019.

15. Skiena S. The Algorithm Design Manual / Steven Skiena. – London: Springer-Verlag, 2008.

16. Portugal R. Quantum Walks and Search Algorithms / Renato Portugal., 2018.

17. Bandara U. AI Game Programming for Beginners / Uditha Bandara., 2012.

18. Heineman G. Algorithms in a Nutshell / G. Heineman, P. Gary, S. Selkow. – Sebastopol: O'Reilly Media, 2009.

19. Goldstone W. Unity Game Development Essentials / Will Goldstone. – Birmingham - Mumbai: Packt Publishing, 2009.

20. Lukosek G. Learning C# by Developing Games with Unity 5.x / Greg Lukosek. – Birmingham: Packt Publishing, 2016.

21. Hocking J. Unity in Action: Multiplatform Game Development in C# with Unity 5 / Joseph Hocking. – New York: MAnning Publications Co., 2015.

22. De Paolis L. Augmented Reality, Virtual Reality, and Computer Graphics / L. De Paolis, P. Bourdot, A. Mongelli., 2017.

23. Lanham M. Learn ARCore - Fundamentals of Google ARCore: Learn to build augmented reality apps for Android, Unity, and the web with Google ARCore 1.0 / Micheal Lanham., 2018.

24. Chowdhury K. Mastering Visual Studio 2017 / Kunal Chowdhury., 2017.

25. Johnson B. Professional Visual Studio 2017 / Bruce Johnson. – Indianapolis: John Wiley & Sons, 2017.

26. Ritchie P. Practical Microsoft Visual Studio 2015 / Peter Ritchie., 2016.

27. Skeet J. C# in Depth / Jon Skeet., 2014.

28. Stellman A. Head First C# / A. Stellman, J. Greene. – Sebastopol: O'Reilly Media, 2013.

29. Mueller J. C# 7.0 All-in-One For Dummies / J. Mueller, B. Sempf, C. Sphar. – Hoboken: John Wiley & Sons, 2017.

30. Розроблення стартап-проекту [Електронний ресурс] : Методичні рекомендації до виконання розділу магістерських дисертацій для студентів інженерних спеціальностей / За заг. ред. О.А. Гавриша. – Київ : НТУУ «КПІ», 2016.

ДОДАТОК

Проеціювання траекторії руху на об’єкти реального світу для мобільного додатку навігації з використанням доповненої реальності

Тези на конференцію «Сучасні проблеми наукового забезпечення енергетики"

УКР.НТУУ”КПІ ім. Ігоря Сікорського” _ТЕФ_АПЕПС_ТВ4197_19М

Аркушів 1

Київ 2019

ПРОБЛЕМА ВИБОРУ РАЦІОНАЛЬНОГО МЕТОДУ ПОЗИЦІЮВАННЯ КОРИСТУВАЧА ДЛЯ СИСТЕМИ НАВІГАЦІЇ

Під час розробки систем внутрішньої навігації у замкнутому просторі постає питання вибору раціонального методу позиціювання користувача в цьому просторі Саме цьому питанню присвячена робота. Ефективність визначається оцінюванням показника «ціна обладнання у відношенні до якості (точності) визначення координат користувача у замкнутому просторі». Розглянемо існуючі методи позиціювання (визначення координат розміщення) користувача у замкнутому просторі.

Система позиціонування в приміщенні, яка базується на BLE (Bluetooth Low Energy), реалізується за допомогою маяків. Маяк - це невеликий електронний пристрій, що складається з мікросхеми та інших електронних компонентів (наприклад, антени) на малій друкованій платі. Маяк - це лише радіопередавач, який посилає сигнал. Цей сигнал може бути захоплений пристроями, які обладнані для його прийому (наприклад, смартфони). Цей метод є енергоефективним, а маяки легко встановлюються. Проте він є відносно «середньо точним» (1 – 3 метри) і до того ж доволі дорогим.

Wi-Fi можна використовувати подібно до маяків BLE, але ця технологія потребує зовнішнього джерела живлення, додаткових витрат на встановлення та дорогого обладнання. На відміну від BLE, сигнал сильніший, і він може покривати більше відстані, збільшуючи точність визначення координат користувача.

Технологія NFC (Near Field Communication) складається з невеликих мікросхем, які не потребують джерела живлення. Пристрій (наприклад, смартфон) виявляє цей чіп і читає його серійний номер, якщо знаходиться в межах 30 см від чіпа. Ця технологія може використовуватися для точного позиціонування в приміщенні в межах 30 см. Ця технологія відрізняється зручністю та швидкістю встановлення з’єднання. Проте ця NFC технологія доступна тільки на нових телефонах, тому для її широкого використання потрібен деякий час.

Існує також технологія Structure from Motion, яка також використовується для систем внутрішньої навігації. Structure from Motion – це техніка комп'ютерного зору для побудови тривимірної (3D) моделі, використовуючи фотографії. Метод SfM використовує зображення або фотографії об'єкта як вхідні дані і в ідеалі створює 3D-модель об'єкта, представленого як набір 3D точок. Після створення карти користувачу надається можливість завантаження фото в систему. Далі на основі зробленої фотографії система визначає її позицію, співвідносить її з картою і повертає користувачу його розраховане місцезнаходження.

Звісно, цей метод не буде найточнішим, а на обробку фотографії і видачі результату потрібен час. Однак він не потребує додаткового обладнання, а лише завантаження додатку на телефон та тому є найдешевшим методом з перерахованих.

Перелік посилань:

Noreikis M. Image Based Indoor Navigation / M. Noreikis. – Espoo, 2014.